VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

NÁVRH, VÝVOJ A IMPLEMENTACE APLIKACE PRO EVIDENCI KULATINY

DESIGN, DEVELOPEMENT AND IMPLEMENTATION OF A ROUND TIMBER EVIDENCE SOFTWARE APPLICATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jiří Štanglica**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Dydowicz, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání diplomové práce

Ústav:	Ústav informatiky
Student:	**Bc. Jiří Štanglica**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Informační management
Vedoucí práce:	**Ing. Petr Dydowicz, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává diplomovou práci s názvem:

Návrh, vývoj a implementace aplikace pro evidenci kulatiny

Charakteristika problematiky úkolu:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrh řešení, přínos práce
Závěr
Seznam použité literatury

Cíle, kterých má být dosaženo:

Cílem práce je, na základě analýzy požadavků firmy, navrhnout, vytvořit a implementovat softwarovou aplikaci pro evidenci kulatiny v architektuře klient-server. Aplikace bude schopna evidovat příjem, manipulaci, pořez a sklad kulatiny, poskytovat přehledy a reporty a bude připravena na možnost rozšíření o mobilní verzi aplikace pro práci v terénu.

Základní literární prameny:

BASL, Josef a Roman BLAŽÍČEK. Podnikové informační systémy. Podnik v informační společnosti. 1. vyd. Praha: Grada, 2008. 283 s. ISBN 978-80-247-2279-5.

MOLNÁR, Zdeněk. Automatizované informační systémy. 1. vyd. Praha: Strojní fakulta ČVUT, 2000. 126 s. ISBN 80-01-02269-2.

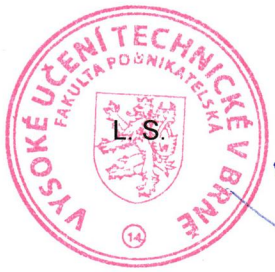
MOLNÁR, Zdeněk. Efektivnost informačních systémů. 1. vyd. Praha: Grada Publishing, 2000. 142 s. ISBN 80-7169-410-X.

ŘEPA, Václav. Analýza a návrh informačních systémů. 1. vyd. Praha: Ekopress, 1999. 403 s. ISBN 80-86119-13-0.

SODOMKA, Petr a Hana KLČOVÁ. Informační systémy v podnikové praxi. 2. aktualiz. a rozš. vyd. Brno: Computer Press, 2010. 501 s. ISBN 978-80-251-2878-7.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17.

V Brně, dne 28. 2. 2017



doc. RNDr. Bedřich Půža, CSc.
ředitel

doc. Ing. et Ing. Stanislav Škapa, Ph.D.
děkan

Abstrakt

Práce se zabývá analýzou potřeb a požadavků dřevozpracující firmy na nový systém vedení skladu dřevní hmoty, a na základě těchto požadavků popisuje proces vývoje tohoto systému. Proces zahrnuje návrh architektury a struktury systému a uložení dat, vývoj samotný, a nakonec také implementaci systému v prostředí firmy včetně zhodnocení ekonomické části procesu. Systém je vyvíjen v programovacím jazyce Javascript a využívá moderní technologie React, Node.js a MongoDB.

Abstract

This thesis is focused on requirements analysis of a sawmill company for a new timber stock evidence system and based on the analysis describes the process of development of the system. The process covers architecture and structure design as well as data storage solution. Apart from the development stage itself, it also covers the implementation of the system in the company's environment and economic evaluation of the process. The system is being developed using the Javascript programming language and uses modern technologies such as React, Node.js and MongoDB.

Klíčová slova

skladový systém, Javascript, React, Node.js, MongoDB, klient, server, aplikace

Key words

stock management system, Javascript, React, Node.js, MongoDB, client, server, application

Bibliografická citace

ŠTANGLICA, J. *Návrh, vývoj a implementace aplikace pro evidenci kulatiny*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2017. 99 s. Vedoucí diplomové práce Ing. Petr Dydowicz, Ph.D.

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 19. května 2017

…..…….……

podpis studenta

Poděkování

Mnohokrát děkuji vedoucímu své diplomové práce, panu Ing. Petru Dydowiczovi, Ph.D. za odborné vedení, jeho čas, ochotu a cenné rady, kterých se mi dostalo při psaní této práce.

Obsah

Úvod

V prostředí dřevozpracujících firem, a zvláště pak firem spadajících do segmentu pilařské výroby je základní surovinou veškerých produktů kulatina – tedy pokácené a odvětvené kmeny stromů. Protože však každá další výroba vyžaduje jiný druh dřeva, jiné rozměry a parametry vstupní suroviny – jednoho kusu kulatiny – je třeba tuto základní surovinu skladovat rozdělenou do patřičných skupin a vést o tomto skladu evidenci. Existují různá řešení, malé závody si s trochou nadsázky vystačí s papírem a tužkou, velcí pilařští giganti pak pracují s rozsáhlými informačními systémy.

Základem každého skladového systému je kromě možnosti kontroly současného stavu skladu také vedení evidence příjmů suroviny do skladu a jejich výdejů ze skladu pro další zpracování. Všechny další funkce jsou navíc a mohou funkcionalitu systému rozšiřovat o všemožné vlastnosti – od funkcí jako je customizace vzhledu systému po možnost zpracování analýz a statistik.

Tato práce se zabývá právě tvorbou a implementací systému pro vedení skladu kulatiny pro malou pilu. V teoretické části budou čtenáři seznámeni s použitými technologiemi a koncepty. Analytická část se bude věnovat současnému stavu ve firmě a procesům, které je třeba v systému pokrýt. Nakonec v části vlastního návrhu řešení bude popsán samotný proces návrhu a vývoje konkrétního systému vedení skladu včetně popisu implementace a ekonomického zhodnocení.

1 Cíl práce a použitá metodika

Cílem této diplomové práce je návrh a zejména vytvoření systému evidence kulatiny pro firmu S T A R P s.r.o, který nahradí současné řešení vedení skladu pomocí tabulek MS Excel. Systém bude pokrývat veškerou současnou funkcionalitu, navíc však přidá funkcionalitu analýz a statistik. Systém bude také připraven na možnost budoucího rozšíření o mobilní aplikace pro ještě vyšší automatizaci procesu vedení evidence příjmů do skladu a pořezu (tedy výdeje ze skladu).

Nejprve bude představena a popsána firma, pro kterou je systém vyvíjen. Poté bude potřeba zanalyzovat procesy, které mají být systémem pokryty a také současné řešení onoho pokrytí. Z těchto analýz vzejdou požadavky na nový systém. Na základě těchto požadavků bude proveden návrh architektury systému, struktury jednotlivých částí, zabezpečení a uložení dat. Poté bude popsán samotný vývoj systému, který bude vyvinut v programovacím jazyce Javascript a bude využívat technologie a komponenty jako je React, Node.js či MongoDB. V další části bude navrhnut a popsán proces implementace systému do firemního prostředí. Nakonec bude rozebrána a zhodnocena ekonomická stránka věci, tedy bude provedena analýza nákladů na realizaci a přínosů nového řešení.

V práci budou využity metody analýz jako je metoda SWOT pro analýzu firmy a současného řešení systému, zkoumané procesy budou zobrazeny pomocí EPC diagramů a současné řešení bude také zhodnoceno metodou HOS8. Při realizaci pak budou využity metody funkčního a datového modelování.

2 Teoretická východiska

V této kapitole budou popsány základní pojmy, se kterými se bude pracovat v dalších částech této práce. Je důležité, aby byly tyto pojmy dobře pochopeny, a proto se tato kapitola zaměří na jejich srozumitelné popsání a vysvětlení. Kde to bude možné a vhodné, budou uvedeny konkrétní příklady pro lepší ilustraci.

2.1 Javascript

Výsledným produktem této práce bude aplikace – systém, vytvořený v jazyce Javascript. Tato kapitola se tedy zabývá vysvětlením pojmu, představením jazyka a jeho stručné historie a příklady využití.

Javascript (též JavaScript či zkráceně JS) je, jak je již z názvu jazyka patrné, je skriptovací jazyk. Ze skutečnosti, že Javascript je skriptovací jazyk plyne, že Javascript je jazyk interpretovaný – není tedy nutné jej kompilovat pro jeho spuštění a vykonání instrukcí. Jeho hlavní využití leží v oblasti webu a internetových stránek a aplikací – umožňuje rozšířit statické HTML stránky o dynamické vlastnosti a učinit je tak interaktivními bez nutnosti znovu načítat celou stránku. Stránky tak mohou reagovat na události (například kliknutí uživatele na tlačítko), měnit svůj obsah či rozložení, kontrolovat zadaná data ve formulářích a mnoho dalšího[1].

Navzdory jménu obsahujícímu slovo Java nemá Javascript s dříve zmíněným jazykem nic společného. Javascript byl vytvořen v roce 1995 společností Netscape (ze které se později vyvinula společnost Mozilla). Ne vždy byl Javascript nazýván Javascriptem – původní název jazyka byl Mocha, krátce po zvolení tohoto názvu došlo ke změně na LiveScript a o dalších několik měsíců později na nynější Javascript. Přejmenování na Javascript však mělo víceméně pouze marketingové účely: v té době byl jazyk Java velmi populární, Javascript tak dostal své jméno proto, aby již názvem zaujal tehdejší potenciální uživatele. Od roku 1996 je Javascript spravován společnosti ECMA, která se až do dnešního dne stará o vydávání a udržování specifikací standardů, jež tvůrci webových prohlížečů využívají pro implementaci Javascriptu do svých produktů. V době psaní této práce je nejrozšířenější iterací standardu ECMA standard nazvaný

[1] TECHOPEDIA INC. What is Javascript (JS)? *www.techopedia.com* [online].

ECMAScript 5, který umožňuje tvůrcům webových stránek využívat moderní prostředky, jako je přístup k médiím, geografické poloze, různým čidlům zařízení (akcelerometr apod.) a servírovat tak uživatelům bohatou paletu možností a funkcionality webových stránek[2].

2.2 Architektura klient – server

Systém vytvořený v rámci této práce se bude skládat ze dvou částí – části serveru a části klienta. Tyto dvě části spolu budou komunikovat prostřednictvím počítačové sítě. Taková architektura či model aplikace se nazývá architektura klient – server. V této kapitole bude tato architektura blíže představena.

Architektura klient – server je, jak již název napovídá, tvořena dvěma vrstvami: klientem a serverem. Server typicky vykonává výpočty a operace s daty, které jsou následně využity klientem. Příkladem z běžného života budiž e-mail nebo webové servery. Každá vrstva, server i klient, využívá vlastní softwarové vybavení. V některých případech může být toto softwarové vybavení stejné pro obě vrstvy, musí však pro každou vrstvu sloužit k jiným účelům. Ve většině případů se však klient a server nachází na různých fyzických strojích (a běží v rámci různých operačních systémů) a tedy i vlastní software je rozdílný[3].

Důležitým požadavkem pro správnou práci architektury klient – server je ustanovení komunikačního protokolu, který klient využívá pro zasílání požadavků na server a komunikačního protokolu, který server využívá k poskytování dat zpět klientovi. Velmi často jsou tyto protokoly součástí jednoho unifikovaného protokolu, jako je například HTTP, TCP, UDP, sokety a podobně[3].

Klient

V tradičním pojetí architektury klient – server označuje pojem klient tu stranu komunikace, jež vznáší požadavky na server a poté zpracovává jeho odpověď. Jakmile klient vznese požadavek, čeká na odpověď serveru tak dlouho, dokud ji nedostane, nebo nedojde k překročení maximální doby čekání. Většina klientů je schopna pracovat naráz s více požadavky, zřídkakdy však s více než čtyřmi. Klient se z pohledu uživatele typicky

[2] WEB EDUCATION COMMUNITY GROUP. A Short History of JavaScript. *www.w3.org* [online]
[3] SOSINSKY B. *Networking bible,* s. 54-55

vyznačuje tím, že obsahuje grafické rozhraní, pomocí kterého uživatelé s klientem pracují[4].

Server

Pojem server můžeme chápat dvojím způsobem: buďto jako určitou aplikaci či softwarové vybavení, které na základě požadavku provede nějaký výpočet, nebo jako fyzický hardware, stroj, na kterém běží software poskytující zmíněné služby. Server typicky nevyvolává komunikaci s klientem (často to ani není v rámci daného protokolu možné), nýbrž čeká na požadavek od klienta, na základě něhož posléze provede požadovanou operaci a klientovi zpět odešle odpověď. Na rozdíl od klientských aplikací nemají serverové aplikace typicky grafické rozhraní a často se k jejich správě využívá rozhraní příkazové řádky[4].

2.3 Technologie využité pro tvorbu serveru

V předchozí kapitole bylo popsáno, co je to architektura klient – server a jaké jsou role jednotlivých stran. Tato kapitola se zaměří na stranu serveru – budou zde popsány technologie a principy využité pro tvorbu serverové části systému, a to včetně softwarové výbavy využité pro správu dat.

2.3.1 Node.js

Na poli webových serverů využívaných napříč internetem existuje několik největších hráčů, kteří jsou využíváni po léta jako ověřená a robustní řešení. Node.js na toto pole vstupuje jako nováček. V době psaní této práce pokrýval Node.js pouze 0,3 % trhu, zatímco například populární Apache je využit v 50,6 % případů[5]. Node.js je totiž v porovnání s ostatními technologiemi mladý produkt. Byl vytvořen v roce 2009 Ryanem Dahlem jako nástroj pro tvorbu vysoce škálovatelných řešení pro tvorbu webových serverů. Samotný Node.js je napsán v jazyce C++ a Javascript a jako interpreta jazyka Javascript využívá V8 Javascript jádro vytvořené společností Google (toto jádro je využito také uvnitř prohlížeče Google Chrome). Propojením nativního C++ a Javascript interpreta vznikla do té doby nevídaná možnost vytvářet serverové aplikace pouze

[4] SOSINSKY B. *Networking bible*, s. 56
[5] W3TECHS. Apache vs. Node.js usage statistics. *www.w3techs.com* [online]

s využitím Javascriptu bez nutnosti kompilace. S příchodem Node.js se Javascript konečně stal plnohodnotným nástrojem k tvorbě webových serverů[6].

Otázka, která vyvstává po přečtení předchozího odstavce zní: proč používat zrovna Node.js a Javascript při tvorbě web serveru? Odpovědí je rychlost a jednoduchost. Na rozdíl od tradičních řešení, jako například již zmíněný Apache, Node.js funguje jinak. Apache a jemu podobná řešení používají pro obsluhu příchozích připojení vlákna. Čím více současných připojení, tím více vláken je tvořeno, což představuje problém – tento přístup není snadno škálovatelný, vlákna konzumují mnoho zdrojů. Node.js naopak všechna připojení řeší v rámci jediného vlákna. V normálním případě by to znamenalo katastrofu – každá časově náročná operace (např. čtení souboru nebo databáze) by znamenala blokování všech ostatních připojení. Node.js však využívá asynchronní neblokující volání a systém událostí a zpětných volání (tzv. callback)[7].

Příkladem budiž čtení dat z databáze. Zatímco tradiční server jako je Apache by postupoval tak, že nejprve zavolá databázové čtení, pak čeká na výsledek, a poté s výsledkem dále pracuje, Node.js zavolá databázové čtení a současně určí callback funkci, která bude vykonána po dokončení databázového volání. Node.js server pak pokračuje dál namísto toho, aby čekal[7]. Příklad v pseudokódu by mohl vypadat takto:

```
// Tradicni pristup
var result = database.query("SELECT * FROM table");
// Cekani na vysledek predchozi operace
foreach(result as row) {
      // ... prace s vysledkem
}
// Dalsi provadeni kodu az po dokonceni predchozich operaci

// Node.js pristup
database.query("SELECT * FROM table", function(result) {
      // ... prace s vysledkem
});
// Dalsi provadeni kodu bez cekani na dokonceni předchozich operaci
```

[6] IHRIG C. *Pro Node.js for Developers*, s. 1
[7] tamtéž, s. 2

2.3.2 REST rozhraní

REST (neboli Representational State Transfer) a architektura či architektonický styl. Byl vyvinut, aby reprezentoval model, který říká, jak by měl moderní web fungovat. REST je ve svém čistém základu nezávislý na konkrétním protokolu, ve skutečnosti však byl vyvíjen v těsné souvislosti s protokolem HTTP. Kvůli této provázanosti s HTTP je vhodné zaměřit se při zkoumání REST architektury na několik charakteristik HTTP, které mají s REST přímou souvislost[8]. Navzdory tomu se v dnešní době mnoho webových služeb a zejména veřejných rozhraní, poskytujících přístup k určité službě, chlubí nálepkou REST, ačkoliv principy a zásady dané architekturou zcela nerespektují[9]. V následujících řádcích tedy budou zmíněné charakteristiky a principy popsány. Konkrétně se jedná o *zdroje, slovesa, identifikátory* a *omezení*.

Zdroje

Cokoliv, co je umístěno na webu je zdroj. Pojem zdroj však prošel vývojem: co v počátcích webu znamenalo pouhý statický dokument či soubor se časem vyvinulo do více abstraktní roviny. Dnes je pojmem zdroj označováno cokoliv, co lze na webu identifikovat, pojmenovat, adresovat či zpracovat. Příkladem budiž obyčejná statická HTML stránka, PDF soubor, videonahrávka či obrázek. Zároveň se však může jednat o abstraktnější věci, jako je například sbírka jiných zdrojů[8].

Slovesa

Pojmem sloveso se zde rozumí metoda HTTP požadavku. HTTP ve své verzi 1.1 definuje sadu sloves, jež značí akci, kterou klient zamýšlí se zdrojem provést. Ze všech definovaných sloves je praktické v souvislosti s REST rozhraním zmínit zejména čtyři z nich: *GET, POST, PUT* a *DELETE*. Tato slovesa a jejich využití lze snadno vysvětlit pomocí srovnání s typickými operacemi s daty v databázi. Databáze (databázová tabulka) typicky obsahuje záznamy; v kontextu REST rozhraní mluvíme o zdrojích. HTTP slovesa zmíněna dříve pak víceméně korespondují s typickými operacemi se záznamy v databázi, takzvanými CRUD operacemi (Create, Read, Update, Delete). Srovnáme-li CRUD

[8] SATERNOS C. *Client-server web apps with JavaScript and Java*, s. 38
[9] PAUTASSO C., WILDE E. a R. ALARCÓN. *REST: advanced research topics and practical applications*, s. 1

operace s HTTP slovesy použitými v kontextu REST rozhraní, dojdeme k následující tabulce:

Tabulka 1 - Srovnání HTTP sloves a CRUD operací
(Zdroj: převzato z [10])

HTTP sloveso	Operace se zdrojem	Analog. databáz. operace
POST	Vytvořit zdroj	INSERT
GET	Získat zdroj	SELECT
PUT	Upravit zdroj	UPDATE
DELETE	Vymazat zdroj	DELETE

Identifikátory

V předchozí části této kapitoly nazvané *Zdroje* bylo řečeno, že zdroj musí být adresovatelný. Tato podkapitola se zabývá právě adresací zdrojů v kontextu REST rozhraní. Zdroje jsou v prostředí webu typicky identifikovány pomocí obecného konstruktu URI (Univorm Resource Identifier), jež bývá dále rozšířen na takzvaný URL (Uniform Resource Locator), jež reprezentuje právě adresu zdroje[11].

V kontextu REST rozhraní jsou zdroje pojmenovávány a identifikovány pomocí cesty v URL adrese. Lomítka v adrese slouží k oddělení zdrojů a vyjádření jejich vzájemného vztahu a hierarchie. REST rozhraní přímo nedefinuje, jak mají být zdroje pojmenovávány, nejlepší praktiky však ukazují, že následující způsob je považován za nejvhodnější: *zdroj* je v cestě URL adresy značen jako podstatné jméno v jednotném čísle; *kolekce zdrojů* pak jako podstatné jméno v množném čísle. Zdroje by měly být značeny malými písmeny, pro oddělení slov je vhodné používat pomlčky[12]. Několik příkladů by mohlo vypadat takto:

```
https://api.example.com/v1/articles
https://api.example.com/v1/article/123
https://api.example.com/v1/article/123/comments
```

První URL označuje kolekci zdrojů *article* (článek). Druhá URL odkazuje na konkrétní článek s ID 123. Třetí adresa pak na kolekci zdrojů *comment* (komentář) pro daný článek nesoucí ID 123. Pomocí HTTP sloves pak lze s danými zdroji pracovat.

[10] SATERNOS C. *Client-server web apps with JavaScript and Java*, s. 39
[11] tamtéž, s. 39-40
[12] tamtéž, s. 40

Omezení

Pro úplnou integraci REST architektury je třeba pamatovat na daná omezení, která jsou pro REST charakteristická. V této kapitole budou krátce popsána tři nejdůležitější omezení.

První omezení říká, že REST architektura je aplikovatelná pro architektury systémů typu *klient – server*[13]. O této architektuře, jejích charakteristikách a vlastnostech tato práce hovoří v kapitole 2.2.

Druhé a nejdůležitější omezení mluví o *bezstavovosti*. To znamená, že server neuchovává žádné informace o relaci s daným klientem. Zjednodušeně řečeno: jakmile server vyřídí požadavek klienta a odešle mu odpověď, zapomene, že spolu kdy komunikovali. Z toho plyne skutečnost, že veškerá data týkající se dané relace a daného kontextu musí uchovávat klient. Ačkoliv tato skutečnost přináší řadu nevýhod – typicky větší požadavky kladené na klientský software či větší objem a četnost přenášených dat – výhody takového přístupu jsou zřejmé. Server je snadno škálovatelný, může lehce využívat technologií distribuce požadavků (tzv. load balancing) a odpadá složitá režie stavu relace a tím pádem se zmenšuje prostor pro chyby[13].

Třetí omezení zčásti vychází z předchozího omezení *bezstavovosti* a zabývá se *vyrovnávací pamětí (cache)*. Jestliže v předchozí části byla zmíněna nevýhoda bezstavovosti ve formě zvýšeného počtu požadavků a přenesených dat, pak právě *cache* je zčásti řešením. Mnoho zdrojů v kontextu rozhraní REST je možno uchovat ve vyrovnávací paměti a v případě nutnosti je odtud opět brát bez nutnosti dotazování se serveru. REST architektura vyžaduje, aby takováto data – zdroje – byla označena, zda je možno je uložit do mezipaměti, nebo ne. Pakliže ano, nese s sebou tato skutečnost značné zvýšení výkonu a rychlosti systému, nicméně také přináší tradiční problém při práci s vyrovnávací pamětí: jak zjistit, zda jsou data v mezipaměti stále ještě platná a nedošlo k jejich změně? Tento problém je však třeba řešit individuálně v rámci konkrétních řešení a je mimo rozsah této práce[14].

[13] SATERNOS C. *Client-server web apps with JavaScript and Java*, s. 41
[14] tamtéž, s 42

2.3.3 Autorizační standard OAuth 2

Jestliže v předchozích kapitolách byly probírány technologie pro tvorbu serverových aplikačních rozhraní (API), které klientským aplikacím umožňují nakládat se zdroji, je nutno ruku v ruce s těmito technologiemi řešit zabezpečení. Je bezpodmínečně nutné, aby přístup k potřebným zdrojům měli pouze příslušní uživatelé a nikdo jiný. Toto zabezpečení lze řešit mnoha způsoby, v posledních letech se však víceméně standardem pro zabezpečení serverových API standard OAuth 2. Tento standard je využíván mnoha velkými hráči na poli webových služeb, zmínit lze například Facebook, Google, GitHub, LinkedIn, PayPal nebo Instagram[15].

OAuth 2 do určité míry vychází ze standardu OAuth 1, v klíčových konceptech se však liší. Mezi OAuth 1 a OAuth 2 však stojí ještě jeden člen: OAuth WRAP (Web Resources Authorization Profiles), představený v roce 2009, který byl vyvinut pro překonání některých limitací standardu OAuth 1. OAuth WRAP však na rozdíl od OAuth 1 nestaví na schématu podpisových certifikátů. Princip fungování byl následující: klientská aplikace přesměruje uživatele na koncový bod serveru společně s klíčem identifikujícím klienta a návratovou URL adresou. Uživatel se pak přihlásí (typicky pomocí webového formuláře) v prostředí serveru (klient tedy neví nic o uživatelově jménu nebo heslu). Jakmile je uživatel úspěšně přihlášen, je přesměrován na návratovou URL adresu společně s autorizačním klíčem. Ihned po tomto přesměrování klient provádí požadavek na server, kde specifikuje obdržený autorizační klíč a zpět obdrží přístupový klíč. Tento klíč pak posílá společně s každým požadavkem na server, což serveru stačí k tomu, aby věděl, že požadavek vychází od autorizovaného uživatele. Důležitou podmínkou je, že veškeré požadavky musí probíhat přes zabezpečený kanál TLS[15].

Tato základní myšlenka delegování pravomoci k přístupu ke zdroji z uživatele na klienta zůstala vlastní i standardu OAuth 2, který je přímým nástupcem OAuth WRAP (OAuth WRAP byl s příchodem OAuth 2 svými tvůrci prohlášen za zastaralý). OAuth 2 tuto myšlenku převzal a dále rozšířil o dva zásadní body: *grant types* (druhy udělení přístupu) a *token types* (druhy klíčů)[16].

[15] SIRIWARDENA P. *Advanced api security: securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE*, s. 91
[16] tamtéž, s. 95

Grant Types

OAuth 2 představil čtyři základní druhy udělení přístupu, které mohou být využity dle druhu klientské aplikace a povahy služby. Jedná se o následující druhy udělení přístupu:

- Authorization Code
- Implicit
- Resource Owner Password Credentials
- Client Credentials

První druh přístupu, Authoriation Code, pracuje přesně tak, jak bylo popsáno ve druhém odstavci kapitoly 2.3.3 a je doporučován zejména pro webové a nativní (zejména mobilní) aplikace. Druh Implicit funguje podobně, je však doporučován zvláště pro Javascriptové aplikace běžící v internetovém prohlížeči. Rozdílem oproti Authroiation Grant metodě se klient nemusí vůči autorizačnímu serveru autentizovat a zpět neobdrží autorizační kód, nýbrž přímo přístupový klíč. Resouce Owner Password Credentials je metoda přístupu, která vyžaduje, aby uživatel věřil klientské aplikaci. Klient využívající tuto metodu, na rozdíl od předchozích dvou, nepřesměrovává uživatele do prostředí serveru, ale nakládá s jeho uživatelským jménem a heslem sám. Po zadání jména a hesla v prostředí klienta se tento dotáže autorizačního serveru na správnost údajů a případně obdrží přístupový klíč. Poslední metodou je metoda Client Credentials. Tato metoda slouží zejména pro automatickou komunikaci dvou strojů, kdy vlastníkem zdroje je právě klientská aplikace. Uživatel v případě této metody nikam nezadává své uživatelské jméno a heslo[17].

Token Types

Jeden z principů představených v OAuth 2 je právě možnost definovat vlastní systém přístupových klíčů. Jediná podmínka je, aby s daným druhem klíče uměly pracovat klientské aplikace. I přes tuto volnost je nejčastěji využívaným druhem takzvaný Bearer Token. Název samotný už napovídá, jak tento klíč funguje: kdokoliv tento klíč vlastní jej může využít pro přístup ke zdrojům – to znamená, že je potřeba dávat na klíč pozor a zajistit, aby jej nikdo nepovolaný nezískal (i proto vyžaduje OAuth 2 komunikaci

[17] SIRIWARDENA P. *Advanced api security: securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE*, s. 96-101

výhradně přes zabezpečený TLS kanál). Bearer Token je libovolný řetězec znaků, který se nejčastěji předává v hlavičce požadavku (hlavička Authorization). Ostatní způsoby zahrnují předání v těle požadavku nebo jako parametr adresy URL[18].

2.3.4 MongoDB

MongoDB je v této práci využita jako databázový stroj pro ukládání dat na straně serveru. Na rozdíl od tradičních databázových systémů, jako je například MS-SQL nebo MySQL se však nejedná o relační databázi. MongoDB patří do skupiny takzvaných NoSQL databází – namísto tradičních tabulek a relací využívá *dokumenty* a *kolekce dokumentů*. Důvodem k opuštění tradičního relačního modelu je zejména snadná možnost škálovatelnosti[19].

Základní myšlenka konceptu dokumentů spočívá v nahrazení tradičního přístupu, kdy v databázi existují tabulky a záznamy jsou reprezentovány řádky jiným, flexibilnějším přístupem – a to právě dokumenty. Dokument je ve své podstatě objekt. A jako objekt může obsahovat kromě tradičních atributů také složitější struktury, jako jsou pole nebo dokonce další dokumenty. Tento přístup pak dokáže lépe reprezentovat komplexní hierarchie a vazby v rámci jediného záznamu. Tato skutečnost je výhodná zejména pro vývojáře aplikací, kteří pracují s moderními objektově orientovanými jazyky. Dalším výrazným rozdílem oproti relačním databázím je skutečnost, že MongoDB nepracuje s pevně daným schématem dokumentu. Tam, kde relační databáze definuje pevnou strukturu tabulky dává MongoDB volnost. Tento přístup je výhodný, jelikož eliminuje nutnost rozsáhlých a náročných modifikací struktury tabulek v případě, že je třeba upravit původní schéma. Místo toho přesouvá MongoDB zodpovědnost za zabezpečení správného nakládání s daty i v případě nového či chybějícího atributu nebo klíče na klientskou aplikaci. Ačkoliv se tento přístup může zdát nešťastný, dává vývojářům volnost v rozhodování, jak budou s rozvíjejícími se modely pracovat[20].

MongoDB však zároveň nepřináší zásadní omezení oproti relačním databázím. Podporuje indexování (včetně indexování vnořených dokumentů), uložené procedury (avšak ve formě Javascriptových funkcí), agregace či metody pro ukládání velkých

[18] SIRIWARDENA P. *Advanced api security: securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE*, s. 102
[19] DIROLF M. a K. CHODOROW. *MongoDB: The Definitive Guide*, s. 1
[20] tamtéž, s. 1-2

souborů. Naopak postrádá některé dobře známé funkce, jako jsou operace typu JOIN nebo pokročilé transakce. Co je však hlavním přínosem MongoDB? Rychlost. MongoDB byla od prvopočátku vyvíjena s ohledem na maximální rychlost a škálovatelnost. Využívá pokročilé a optimalizované způsoby komunikace s databázovým strojem, ukládání výsledků v mezipaměti, automatickou optimalizaci dotazů, prediktivní vyhrazování místa na disku pro budoucí rozšíření datových struktur a další. Přes toto všechno a přes snahu o zachování co nejvíce vlastností relačních databází není MongoDB schopno (avšak ani navrženo) dělat všechno to, co umí relační databáze. Filozofický model práce s MongoDB spočívá spíše v přenášení co nejvíce logiky a výpočtů na stranu klienta. Tím pádem se databázový stroj zabývá opravdu pouze operacemi s čistými daty, a to je důvodem tak vynikající rychlosti MongoDB[21].

2.4 Technologie využité pro tvorbu klienta

V této kapitole budou popsány technologie a principy využité při tvorbě klientské části systému. Stejně jako serverová část je i klient napsán v jazyce Javascript a využívá moderní technologie, které v posledních letech zažívají rostoucí popularitu.

2.4.1 Single Page Aplikace

Klientská část systému je vytvořena v duchu architektury Single Page Aplikace (SPA). Tato architektura pracuje s myšlenkou jediné HTML stránky, která slouží jako obal pro veškeré dění v aplikaci. Všechny stránky (či obrazovky) aplikace jsou vykresleny v rámci toho to jediného HTML dokumentu a veškerou interakci s uživatelem ovládá Javascript. Z této skutečnosti plyne, že při vývoji SPA je téměř všechna práce odvedena na klientské (front end) části. SPA svým charakterem do značné míry připomínají tradiční nativní aplikace dané platformy, a to jak způsobem používání uživateli, tak přístupem k vývoji těchto aplikací. Rozdílem však je, že SPA běží v rámci internetového prohlížeče, zatímco nativní aplikace běží v rámci vlastního procesu[22]. Pro lepší pochopení výhod SPA je dobré srovnat architekturu a fungování tradičních webových aplikací, nativních aplikací, a právě Single Page Aplikací.

[21] DIROLF M. a K. CHODOROW. *MongoDB: The Definitive Guide*, s. 3
[22] FINK G. a I. FLATOW. *Pro Single Page Application Development*, s. 3

Tradiční webové aplikace

Tento model fungování internetových stránek a webových aplikací byl představen již v 90. letech minulého století a základní principy fungování jsou dodnes nezměněny. Tento druh aplikací je silně serverově orientovaný. To znamená, že uživatel nemusí nic instalovat a aktualizovat (kromě internetového prohlížeče), což přináší jasnou výhodu pro vývojáře, kterým stačí aktualizovat kód na straně serveru. Tradiční webové aplikace přenášejí pomocí HTTP požadavků a odpovědi vždy celou HTML stránku a akce uživatele (odeslání formuláře, navigace na jinou stránku atd.) vždy vyvolají překreslení celého obsahu. Taková funkcionalita není příliš přívětivá z uživatelského hlediska, jelikož uživatelé musí často čekat na překreslení stránky, nezůstane uložený předchozí stav a podobně[23]. Postupem času došlo k vývoji nových technologií, zejména hovoříme o Ajaxu, které umožnily tento tradiční přístup posunout blíže interaktivnějším nativním aplikacím. Stále se jedná o serverově orientované aplikace, které vyžadují překreslování při většině uživatelských akcí (zejména navigaci). Některé z nich, příkladem budiž odeslání formuláře, však lze nyní řešit pomocí Ajax (asynchronní HTTP požadavky) požadavků. Toto přineslo zlepšení uživatelské zkušenosti, která se přesunula o něco blíže nativním aplikacím[24].

Nativní aplikace

Nativní aplikace jsou samostatné aplikace, které jsou spustitelné pouze v rámci určité platformy (např. pouze Windows nebo pouze macOS), a je většinou je nutné je nejprve nainstalovat. Omezení na danou platformu, pro kterou byly vytvořeny, je jejich slabost i silná stránka: nelze je používat v rámci jiné platformy, naopak ale umožňují využití všech předností dané platformy, jako například hardwarovou akceleraci, přímý přístup k periferiím nebo senzorům zařízení či dalších vestavěných vlastností daného operačního systému. Oproti webovým aplikacím je v tomto případě značně složitější režie distribuce aplikace a následně jejích aktualizací. Naopak uživatelská zkušenost a pohodlnost používání nativních aplikací ty webové na míle předhání. Aplikace jsou typicky rychlé, umožňují zachovat rozpracovanou práci (takzvaný lokální stav) a zejména ve většině případů dokáží pracovat i v režimu bez přístupu k internetu[25].

[23] FINK G. a I. FLATOW. *Pro Single Page Application Development*, s. 6-7
[24] tamtéž, s. 9
[25] tamtéž, s. 10-11

Single Page Aplikace

Předchozí dva odstavce přinesly srovnání tradičních webových aplikací s nativními aplikacemi. Tento odstavec se zaměří na to, kam mezi tyto dva přístupy patří SPA. Jak již bylo řečeno v úvodu této kapitoly, SPA pracuje s jedinou HTML stránkou, která s využitím Javascriptu tvoří veškeré prostředí aplikace. V SPA nikdy nenastává plné překreslení stránky s novým HTML obsahem ze serveru, navigace nevyvolá opětovné načtení celé aplikace. Veškeré objekty v rámci dané HTML stránky jsou dynamicky vytvářeny a nahrazovány v rámci životního cyklu SPA prostřednictvím vestavěných mechanismů směrování a šablon. Díky absenci překreslování a opětovného načítání stránek lze také využít mechanismů vnitřní paměti prohlížeče, případně lokálního úložiště dat pro zachování stavu aplikace (například zachování nedokončené práce). Zároveň, v případě dobrého návrhu SPA, lze také do určité míry zajistit schopnost práce v offline režimu, kdy data jsou ukládány a načítány z lokálního úložiště a synchronizována se serverem po obnovení připojení. Stejné principy jsou využity v případě, že uživatel zavře okno prohlížeče a později aplikaci opět otevře[26]. Pro porovnání všech tří přístupů a jejich vlastností lze využít následující tabulku:

Tabulka 2 - Porovnání vlastností jednotlivých druhů aplikací
(Zdroj: převzato z [27])

Vlastnost	Tradiční	Nativní	SPA
Podpora více platforem	Ano	Ne	Ano
Uchování stavu aplikace	Ne	Ano	Ano
Není vyžadována instalace	Ano	Ne	Ano

Z tabulky je patrné, že SPA leží mezi tradičními webovými aplikacemi a nativní aplikacemi. Z každého světa přebírá hlavní charakteristiky a nejlepší vlastnosti, zatímco nevýhody do značné míry potlačuje.

2.4.2 React

Klientská část aplikace, která bude vytvořena v rámci této práce, bude vytvořena v jazyce Javascript a pro tvorbu je využita knihovna React. Tato kapitole se zabývá právě popisem Javascriptové knihovny React, jejích výhod a vlastností.

[26] FINK G. a I. FLATOW. *Pro Single Page Application Development*, s. 11-12
[27] tamtéž, s. 11

React je javascriptová knihovna, lze na něj však nahlížet také jako na javascriptový framework. Původ Reactu lze hledat ve společnosti Facebook, v oddělení, které mělo na starost tvorbu, správu a distribuci reklam. React zde vznikl jako náhrada tradičního MVC frameworku, aby vyřešil problém budování komplexních klientských uživatelských rozhraní, kde se operační data mění v průběhu času. Tento problém nemá triviální řešení, zvláště při uvážení potřeby možnosti škálovatelnosti takové aplikace na úroveň velikosti Facebooku. React změnil způsob, jakým lze takové aplikace vyvíjet. Pro veřejnost byl React uvolněn v roce 2013[28].

Filozofie práce a vývoje v Reactu „hází rukavici" tradičním konvencím a postupům, které léta platily za dané při vývoji klientských javascriptových aplikací. React oproti těmto běžným postupům představuje mnoho nových paradigmat a jiný pohled na vývoj. React přináší sadu nástrojů, postupů a konceptů, které při vhodném použití vývojářům značně usnadní vývoj Single Page Aplikací. Tato kapitola nejdůležitější z nich představí; jedná se zejména o virtuální DOM, JSX či koncept datového toku Flux[28].

Co tedy React přináší jiného oproti tradičním MVC frameworkům, jako je například Angular.js? React není jen další z MVC frameworků. Facebook se nevydal cestou již jednou objeveného kola – místo toho se soustředil na jednu jedinou funkcionalitu, a to je zobrazování dat, která se mění v čase, v uživatelském rozhraní. Ačkoliv tento problém byl již také vyřešen dříve, React se zaměřuje na zvládnutí tohoto problému způsobem, který lze snadno udržovat, škálovat a testovat. Uživatelská rozhraní, která pracují s daty, jež se mění v čase, jsou dnes naprosto běžná věc. Jak již bylo popsáno v předchozích kapitolách této práce, trend dnešní doby je přenechávat (čím dál větší) část práce klientům – tedy typicky webovým prohlížečům. S tímto přístupem však vzniká problém: jak přibývá další a další funkcionalita, kód se stává čím dál méně udržitelným. Kód pro novou funkcionalitu se „lepí" na různá místa a vzniká zmatek. Přijde-li potřeba restrukturalizace aplikace, je velmi náročné tuto operaci zvládnout tak, aby se celá aplikace nerozsypala. V mnoha případech je toto slabé místo i tradičních MVC frameworků, kde existuje úzká vazba mezi modelem, který drží stav dat (M), pohledem, který je vykreslen uživateli (V) a controllerem, který předchozí dva propojuje a emituje

[28] GACKENHEIMER C. *Introduction to React*, s. 1

události pro změnu dat nebo jejich zobrazení (C)[29]. React se proto nestal jen dalším MVC frameworkem. Ve skutečnosti je React ve své základní formě pouze ono V z celé architektury, tedy View – pohled. React tedy slouží k popisu uživatelského rozhraní a přináší mechanismus, jak toto rozhraní změnit v případě změny dat. React je tvořen deklarativními komponentami, které popisují uživatelské rozhraní. Komponenty lze snadno kombinovat a tvořit tak složitější uživatelské prvky a celé stránky[30]. V následujících podkapitolách bude popsáno, co jsou to ony komponenty a jaké mechanismy a koncepty React implementuje pro řízení změny dat a zobrazení oněch komponent.

Komponenty

Komponenta je jádro celého Reactu a každé aplikace s jeho pomocí vytvořené. Komponenty tvoří základní stavební bloky aplikace, které pak tvoří složitější struktury. Filosofie tvorby aplikace z malých, navzájem izolovaných stavebních bloků je základ snadného škálování aplikace takto tvořené[31]. Každá komponenta je tvořena JSX reprezentací zobrazení (viz další podkapitoly) a volitelně dalšími metodami a vlastnostmi (více detailů lze nalézt v [32]). Příkladem takové komponenty budiž následujících několik řádků kódu (zápis využívá Javascript syntaxi ve verzi ES6, více dále):

```
class Hello extends React.Component {
    render() {
        return <p>Hello world!</p>;
    }
}
```

Takto vznikne jednoduchá komponenta, která pouze vykreslí element P obsahující text „Hello world!“. Nyní je cílem vypsat tento text třikrát za sebou. Využijeme proto kombinování komponent a vytvoříme následující:

```
class Greetings extends React.Component {
    render() {
        return
        <div>
            <Hello /><Hello /><Hello />
        </div>;
    }
}
```

[29] GACKENHEIMER C. *Introduction to React*, s. 2-3
[30] tamtéž, s. 4
[31] tamtéž, s. 28
[32] FACEBOOK Inc. Components and Props – React. *https://facebook.github.io/react/* [online]

Tímto jednoduchým způsobem lze využít již existující komponentu `Hello` v nové komponentě `Greetings`. Na tomto principu pak stojí celá filosofie škálovatelnosti aplikací vytvořených pomocí Reactu.

Virtuální DOM

Virtuální DOM je jeden z nejdůležitějších konceptů celé knihovny React. Prvotní verze Reactu při každé změně dat a vypočítání nového stavu a požadavku na překreslení uživatelského rozhraní vždy překreslily celou stránku, všechny komponenty. To však nebylo efektivní, a navíc to způsobovalo problikávání komponent v průběhu překreslení a další. React tedy přišel s konceptem virtuálního DOMu, kdy při změně dat a požadavku na překreslení nedojde k zásahu do celého uživatelského rozhraní a překreslení všech komponent, ale namísto toho je využit virtuální DOM, pomocí kterého je vypočítána minimální nutná úprava skutečného DOMu, a jsou tedy překresleny jen ty komponenty, které byly změnou dat ovlivněny, což s sebou přináší značné zvýšení rychlosti běhu a odezvy celé aplikace[33].

JSX

V podkapitole Komponenty byla zmíněna reprezentace JSX. JSX je transformační vrstva, která transformuje syntax velmi podobný XML, jež je využit při tvorbě komponent, na syntaxi, které rozumí vnitřní mechanismy Reactu a Javascriptu pro samotné vykreslování HTML prvků na stránku. Použití JSX při práci s Reactem není vyžadováno, je však silně doporučeno, jelikož značně zjednodušuje zápis. Jak JSX pracuje lze ukázat na následujícím příkladě[33]:

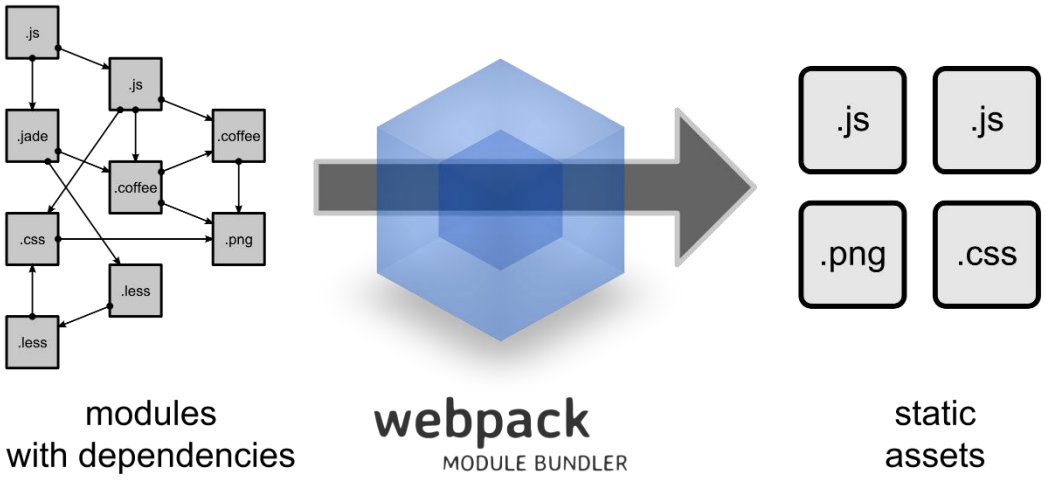
```
// Zapis JSX
React.render(
    <div>
        <h1>Header</h1>
    </div>
);
// Preklad JSX na Javascript
React.render(
    React.createElement('div', null,
        React.createElement('h1', null, 'Header')
    );
);
```

[33] GACKENHEIMER C. *Introduction to React*, s. 17

V této kapitole byl představen framework/knihovna React ve své naprosto základní formě. Při skutečné práci do vstupují do hry další knihovny, middleware prostředníci a další a další nástroje a koncepty, které jsou však mimo rozsah této práce. Některé z nich však budou zmíněny a přiblíženy kapitole 4.

2.4.3 Webpack

V předchozí kapitole bylo popsáno, co to je React a jak funguje, a také to že jeho základním stavebním kamenem je komponenta. Jelikož aplikace se typicky skládají z desítek až stovek komponent, bylo by nesmírně obtížné až nemožné všechny tyto komponenty ručně referencovat a importovat do kódu například pomocí `script` tagu. Proto je v této práci – a při práci s Reactem obecně – využit nástroj Webpack. Webpack je nástroj, který z mnoha vstupních souborů (modulů), které jsou navzájem provázány a závisí na sobě, generuje jeden výstupní soubor, jeden balíček (z anglického bundle, kdy Webpack je označován jako module bundler). Výstupní soubor (či více souborů, v závislosti na konfiguraci) je pak statickou reprezentací oněch modulů a jejich vazeb[34]. Následující grafika proces vizualizuje:



Obrázek 1 - Proces generování balíčku z modulů
(Zdroj: [35])

[34] SCOTCH.io. Getting Started with Webpack: Module Bundling Magic, *https://scotch.io* [online]
[35] WEBPACK. webpack, *https://webpack.github.io/* [online]

Síla Webpacku leží zejména v takzvaných „loaderech“, což jsou přídavné moduly sloužící k rozšíření základní funkcionality Webpacku. Díky loaderům umí Webpack kromě čistého Javascriptu, který je zahrnut v jádře samotného Webpacku, pracovat i s dalšími formáty souborů, jako jsou kaskádové styly, nebo jejich preprocesory (SASS, LESS), obrázky a další. Protože téměř žádná webová aplikace se bez stylů, obrázků a dalších zdrojů neobejde, je použití loaderů nutné[36].

Jeden z nejdůležitějších loaderů použitých v této práci je `babel-loader`. Ten umožňuje tvořit kód aplikace s využitím nejnovějších možností Javascriptu, tedy verze ES2015 (neboli ES6). Jelikož tato verze standardu Javascriptu ještě není většinou internetových prohlížečů podporována (podporovaný standard je ES5), je potřeba použít překladač, který skripty vytvořené s použitím ES6 přeloží do starší verze ES5, což zajistí hladké fungování aplikace napříč prohlížeči. K tomu právě slouží nástroj Babel[37], jež je importován do Webpacku v podobě loaderu, a který v průběhu procesu generování balíčku ze vstupních modulů nejprve přeloží všechny skripty z verze ES6 do verze ES5, a teprve pak vytvoří finální balíček. Příklad takového překladu může vypadat následovně:

```
// ES6 zapis
[1,2,3].map(n => n + 1);

// Babel...

// ES5 preklad
[1, 2, 3].map(function (n) {
    return n + 1;
});
```

[36] SCOTCH.io. Getting Started with Webpack: Module Bundling Magic, *https://scotch.io* [online]
[37] BABEL. Babel, https://babeljs.io/ [online]

3 Analýza současného stavu

V této kapitole se budu věnovat představení firmy, pro kterou budu vytvářet a implementovat aplikaci. Budou zde představeny procesy ve firmě a blíže specifikovány procesy, které budou vytvářenou aplikaci pokryty. Také zde budou představeny požadavky firmy na aplikaci a provedeny analýzy, na jejichž základě budu navrhovat vhodné řešení.

3.1 Představení firmy

3.1.1 Firma S T A R P s.r.o.

Název: S T A R P s.r.o.

Právní forma: Společnost s ručením omezeným

Založeno: 8. 3. 1991

Počet zaměstnanců: 18

Firma S T A R P s.r.o. je dřevozpracující závod, který působí na českém (a mezinárodním) trhu již více než 20 let. Jedná se o malou firmu se sídlem a výrobou na Bruntálsku, ve městě Rýmařov, která však expeduje své výrobky nejen po celé České republice, ale také do ostatních zemí – konkrétně Rakouska a Belgie. Firma se primárně zabývá zpracováním surové dřevní hmoty (neboli kulaté či kulatinové dřevo). Zpracování kulatiny zahrnuje především produkci latí a hranolů různých rozměrů podle přání zákazníka, dalším zpracováním těchto latí nebo bočního řeziva (např. přesným rozřezáním na malé a tenké hranolky o tloušťce do 40 mm) a produkcí paliva (zbytkové a boční řezivo, které nelze dále zpracovat), které je dále buďto přímo prodáváno, většinou však zpracováno na štěpku (biomasa).

Firma spolupracuje s některými významnými společnostmi působícími v České republice a v Rakousku, přičemž nejvýznamnějším obchodním partnerem je mezinárodně působící společnost SECA Borohrádek (Serafin Campestrini s.r.o.) – významný výrobce palubek v České republice a v Rakousku (SECA Ottensheim). Zároveň firma exportuje dřevěné hranolky do Belgie, kde spolupracuje s obchodním partnerem Infra-Wood BVBA, který zprostředkovává prodej pro různé stavební firmy po celé Belgii. Průměrně

dva až tři kamiony čerstvého nebo impregnovaného dřeva (ošetřeného proti plísni a škůdcům během přepravy a skladování) odjíždí do Belgie každý týden.

V předminulém roce firma získala dotaci programu Ministerstva pro místní rozvoj, díky které celý minulý rok probíhaly rekonstrukce výrobních prostor, a zvláště pak pořizování nových technologií na výrobu dřevěných peletek, které se stávají čím dál populárnějším palivem do kotlů v domácnostech a organizacích. V souvislosti s tím také firma přijala několik nových zaměstnanců z řad místních občanů – firma tak působí v oblasti s jednou z nejvyšších hladin nezaměstnanosti v České republice k – na místní poměry – významnému zaměstnavateli.

Firma je držitelem certifikátu kvality *Chain of Custody* společnosti *PEFC*, který zajišťuje a zavazuje firmu k dodržování pravidel obchodu s dřevní hmotou. Certifikát dokazuje, že firma nepracuje s dřívím, které by mohlo pocházet z kontroverzních zdrojů (aktivity, které např. nesplňují požadavky místní, národní nebo mezinárodní legislativy, používají geneticky modifikované organismy, přeměňují lesy na jiné vegetační typy atd.).

3.1.2 Právní skutečnosti

Firma je zapsána a vedena u Krajského soudu v Ostravě pod položkou C 369, k zápisu došlo 8. března 1991. Identifikační číslo firmy je 14612755. Firmu založili tři společníci, v současnosti však již vystupují pouze dva – Jiří Štanglica a Ing. Zdeněk Adam, oba s polovičním podílem a vkladem ve výši 60 000 Kč. Jednatele firmy je Jiří Štanglica, který za společnost jedná samostatně. Předmětem podnikání podle živnostenského zákona je *výroba, obchod a služby neuvedené v přílohách 1 až 3 živnostenského zákona*.

3.2 Mise a vize firmy

3.2.1 Mise

Misí firmy S T A R P s.r.o. je poskytovat svým zákazníkům širokou škálu výrobků dřevozpracující sféry, zvláště pak dřevěných latí a hranolů, přesně nařezaných a kvalitativně ošetřených. Cílem firmy je poskytovat konzistentní kvalitu výstupů a efektivní zvládání procesů, zároveň však být flexibilní a umět rychle a efektivně reagovat

na požadavky a přání svých zákazníků. Misí firmy S T A R P s.r.o. je držet stálou, vysokou kvalitu svých služeb a být svým zákazníkům spolehlivým obchodním partnerem.

3.2.2 Vize

Vizí firmy S T A R P s.r.o. je stát se svým zákazníkům symbolem jistoty v obchodování, příjemným partnerem v byznysu i mimo něj. Jméno S T A R P by mělo být pozitivně vnímáno nejen partnery v obchodním styku a zákazníky, ale také občany okolních obcí, vnímáno jako dobrý soused, který působí jako dobrý zaměstnavatel, a který rád pomůže komukoliv s jeho potřebami.

3.3 Nabízené služby

3.3.1 Služby pro firmy

Obchodování s ostatními podniky – B2B obchod – tvoří jádro obchodní činnosti firmy S T A R P s.r.o. Firma obchodním partnerům nabízí tyto služby dřevozpracující výroby:

- Pořez kulatiny do průměru 35 cm – technologie kombinace rámové a rozmítací pily
- Produkty smrkového řeziva
    - Hranoly o rozměrech 50 až 100x100 mm až 80 až 100x120 mm
    - Fošny o rozměrech 50x160 mm a 50x200 mm
    - Latě o rozměrech 30x50 mm
    - Prkna o rozměrech 22x100 až 180 mm a 32x150 nebo 180 mm
- Pilařský výrobní odpad
    - Piliny
    - Barevná štěpka

S prvním bodem seznamu souvisí také poptávka firmy po vstupní surovině, tedy po kulatinovém dřevě. Firma poptává jehličnaté řezivo – smrkové, které nakupuje od lokálních dodavatelů.

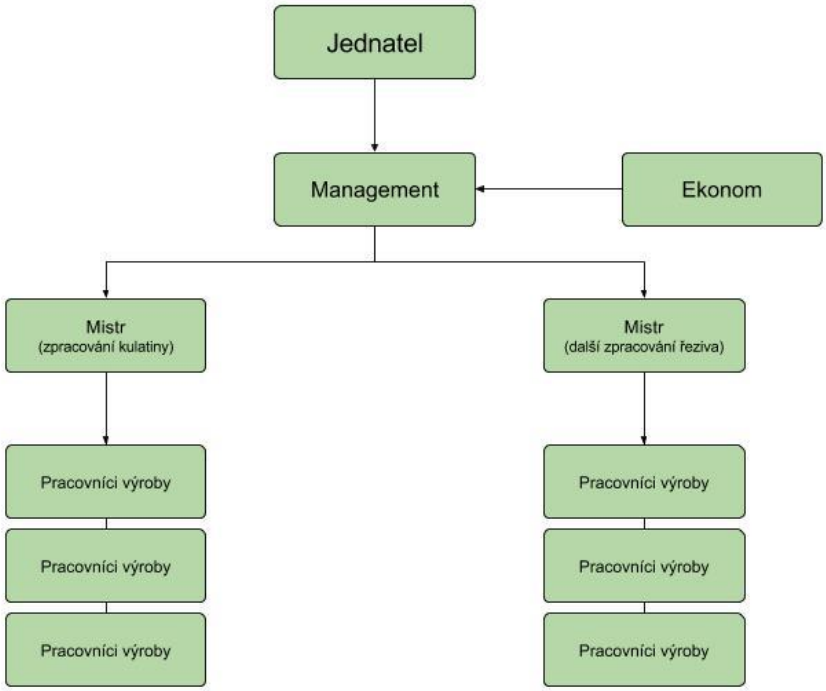
3.3.2 Služby pro spotřebitele

Firma nabízí spotřebitelům – fyzickým osobám, občanům – své výrobky stejně tak jako je nabízí partnerům v obchodním styku. Řeč je zejména o deskách či hranolech,

které lidé využívají pro vlastní stavební účely – ploty, bednění, boudy, zahradní chatky, střechy a podobně. Zároveň lidé od firmy často kupují produkty zbytkové výroby – zejména se jedná o piliny či štěpku (biomasu), které lidé využívají pro topení, údržbu a dekoraci zahrad a podobně.

3.4 Organizační struktura

Společnost S T A R P s.r.o. je spíše malá firma čítající okolo 18 zaměstnanců. Z toho dva jsou pracovníci managementu a dále dva mistři, pod které spadají ostatní zaměstnanci podle oddělení (oddělení zpracování kulatiny a oddělení dalšího zpracování dřeva). Organizační struktura společnosti je zobrazena na diagramu níže.



Obrázek 2 - Organizační struktura společnosti
(Zdroj: Vlastní zpracování)

3.5 Informační technologie

3.5.1 Hardware a síť

Jelikož firma S T A R P s.r.o. je relativně malá společnost, nevyužívá mnoho různorodého hardware pro zajištění každodenního běhu firmy – nepočítáme-li stroje

využívané pro samotnou výrobu a zpracování dřeva. Stroje totiž nejsou žádným způsobem napojeny na firemní síť a informační systémy firmy, a to zejména proto, že velké množství strojního vybavení firmy (rámová pila, hoblovka, omítací pila, kapovací pila) pochází z období 70. až 90. let minulého století.

Firma tedy disponuje pouze několika kusy hardware, který je propojen v rámci firemní sítě. Jedná se o následujícími prvky hardware:

- Dva počítače plus periferie
- Multifunkční tiskárna
- Dva IP telefony
- Síťové prvky
    - ADSL Wi-Fi router
    - 10portový přepínač
    - Kabeláž

Počítače byly pořizovány a obměňovány postupně, mají proto rozdílnou konfiguraci a rozdílný výkon. Oba počítače jsou připojeny do firemní sítě a uživatelé mezi sebou sdílí jak soubory, tak programové vybavení (viz kapitola 3.5.2). Tiskárna je taktéž připojena do firemní sítě, a tedy k dispozici oběma počítačům. Dřívější klasické telefony připojené klasickou telefonní linkou nahradily v minulých letech IP telefony. Internet a telefonní linka je poskytována společností O2 Czech Republic a.s. Stejná společnost také dodala ADSL Wi-Fi router.

Firma nedávno provedla výrazný upgrade firemní sítě s účelem rozšíření možností stávajícího zázemí. Původní stav byl takový, že firma disponovala pouze jednoduchým routerem a několika síťovými kabely nataženými k počítačům. Nyní byla síť přeprojektována tak, že router a přepínač jsou umístěny v rozvaděči, ze kterého jsou vedeny kabely ke stanicím. Rozvaděč disponuje dalším prostorem, který bude v budoucnosti využit pro rozšíření síťové infrastruktury. Dále je plánováno propojení hlavní budovy výroby, ve které se nachází kancelář a místnost s rozvaděčem, a vedlejší budovy, kde probíhá další zpracování dřeva. Propojení bude realizováno optickým kabelem. Toto rozšíření bude následováno pořízením serveru a vybudováním

kamerového systému s možností online monitoringu prostorů, a to zejména z důvodu relativně častých drobných krádeží.

3.5.2 Software

Firma nevyužívá žádný ERP systém. Vzhledem k velikosti firmy, počtu zákazníků a zakázek to vedení firmy nepovažuje za nutné. Správa zákazníků, kontaktů atd. tak přechází na kombinaci papírových záznamů, adresáře e-mailového klienta a případně údajů z účetního software a podobně. To jistě není optimální stav, nicméně vzhledem k velikosti firmy je relativně udržitelný.

Společnost S T A R P s.r.o. tedy v současnosti využívá ke každodenní práci zejména produkty firmy Microsoft a balíčku MS Office. Oba počítače jsou vybaveny operačním systémem Windows 7, balíčkem MS Office verze 2010 a jako e-mailový klient je využíván Microsoft Outlook 2010.

Pro vedení účetnictví je využíván program POHODA od společnosti STORMWARE s.r.o. ve své základní verzi a licenci pro jednu pracovní stanici. Personalistiku a mzdové hospodářství zajišťuje program DUNA MZDY od společnosti TILL CONSULT a.s.

Agendu dřevařské výroby, plánování pořezu, vystavování přejímek a dalších věcí spojených s dřevařskou výrobou využívá firma S T A R P s.r.o. několik modulů specializovaného systému MELCO Dřevařský SW od firmy MELCO spol. s.r.o. Firma tento software využívá již od dob jeho verze pro systémy DOS, která obsahovala tradiční DOSové grafické rozhraní (i po portu na systémy Windows XP a vyšší). V současné verzi je již systém přepracován do standardní podoby programů pro systém Windows. Firma S T A R P s.r.o. byla jedním z prvních zákazníků, kteří na novou podobu programu přešla, a podílela se tak na testování a ladění programu MELCO Dřevařský SW.

Firma má také nastavená pravidla pro zálohování vitálních dat, jako jsou databáze účetnictví, personalistiky a některé dokumenty každodenní operativy, jako jsou přejímky, faktury, objednávky a další. Zálohy jsou prováděny automaticky jednou denně na externí hard disk. K této operaci je využívána softwarová aplikace Cobian Backup od autora Luis Cobian.

3.6 Činnosti a procesy podniku

3.6.1 Stručný přehled procesů

Ve firmě S T A R P s.r.o. probíhá několik standardizovaných procesů spojených s výrobní činností podniku, které pokrývají veškeré potřeby každodenního provozu. Samozřejmě ve firmě probíhají i procesy mimo oblast dřevařské výroby, jako například procesy týkající se zpracování náboru zaměstnanců, vyplácení mezd a další, ale tento druh procesů nebudu v této práci detailněji rozebírat a popisovat. Procesy související s výrobou tedy lze rozdělit takto:

1. Hlavní procesy související s výrobou
    - Kontrola stavu skladových zásob surové hmoty
        - Každodenní kontrola stavu skladu kulatiny, objednání nových dodávek kulatiny od různých dodavatelů.
    - Příjem surové hmoty, manipulace, třídění v rámci skladu kulatiny
        - Složení nákladního vozu na příjmové místo, klasifikace třídy, změření rozměrů a označení kulatiny, manipulace, evidence manipulované kulatiny, roztřídění na dané místo ve skladu podle označení.
    - Zásobování výroby kulatinou
        - Přesun kulatiny ze skladu na zásobník dodávacího zařízení do výrobní haly, zajištění plynulé dodávky kulatiny výrobě.
    - Zpracování jednotlivých kusů kulatiny na výrobní lince
        - Evidence přijatých kusů kulatiny do výroby, posun kulatiny do rámové pily, oddělení středového a bočního řeziva, přesun středové části do rozmítací pily, bočních částí do omítací pily nebo sekačky.
    - Příprava produktů výroby pro přesun do skladu dřevní hmoty
        - Rozdělení latí a fošen dle rozměrů, skládání do kostek a vázání, označení kostek, přesun na místo pro převoz do skladu dřevní hmoty.
    - Zásobování následné výroby dřevní hmotou ze skladu dřevní hmoty
        - Přesun kostek latí požadovaných rozměrů ze skladu dřevní hmoty na místa zkracování.

- Zkracování latí na požadovanou délku a přesun do meziskladu
  - Přesné zkracování latí na požadované délky, skládání do kostek, označování kostek a přesun kostek do místa meziskladu pro další výrobu.
- Zpracování zkrácených latí na pile/frézce a paketizace
  - Zpracování zkrácených latí na latě a přířezy o tloušťce do 40 mm, skládání a vázání latí a přířezů do paketů, skládání paketů do kostek a vázání, příprava k odvozu k impregnaci nebo do skladu dřevní hmoty.
- Impregnace kostek latí a přířezů
  - Přesun kostek latí a přířezů k impregnační vaně, impregnace a okapání a přesun na místo odvozu do skladu dřevní hmoty.
- Přesun kostek hranolků do skladu dřevní hmoty
  - Přesun kostek latí a přířezů na určené místo ve skladu dřevní hmoty, případně příprava místa ve skladu, evidence stavu skladu.
- Expedice ze skladu a nakládání nákladních vozů
  - Expedice kostek latí a přířezů ze skladu dřevní hmoty, nakládání kostek na nákladní vozy podle specifikace objednávky.

2. Vedlejší procesy související s výrobou
   - Kontrola stavu zásobníku pilin
     - Denní kontrola stavu zásobníku pilin, objednání odvozu pilin, příprava zásobníku na vyprázdnění do nákladního vozu odběratele.
   - Kontrola stavu skládky biomasy
     - Denní kontrola stavu skládky biomasy, objednání odvozu biomasy, nakládání biomasy do nákladního vozu odběratele.

3. Procesy související s obchodem
   - Objednávání dodávek surové dřevní hmoty
   - Objednávání nákladních vozidel pro převoz výrobků k zákazníkům
   - Zpracování požadavků na výrobu – specifikace pořezu, výroby atd.
   - Tvorba přejímacích protokolů, fakturace vyrobeného a expedovaného zboží

3.6.2 Pokrytí procesů informačními systémy

Firma S T A R P s.r.o. je malá firma, která se zaměřuje na „tvrdou“ práci založenou na zpracování hmoty robustním strojním vybavením. Toto vybavení je z velké části relativně staré – některé kusy strojní výbavy byly vyrobeny v 60. a 70. letech minulého století a postupem času prošly jen opravami či drobnými vylepšeními. Automatizace, on-line sběr dat ze strojů či jakékoliv jiné propojení s IT systémy či počítačově řízené sbírání dat o výrobě je tedy v současné situaci nemožné. Zbývá tedy zapojit prostředky IS zejména do procesů typu evidence, plánování výroby, plánování dodávek hmoty do skladu a tak podobně – obecně veškerých procesů, které probíhají na vyšší organizační úrovni, zejména mimo samotnou výrobní halu – v kanceláři s využitím stolních počítačů.

Z procesů zmíněných výše jsou tedy informačními systémy pokryty zejména procesy týkající se obchodu. Výrobní procesy popsané výše lze pokrýt informačním systémem relativně složitě, avšak do určité míry to možné je. V současném stavu firmě chybí zejména důkladné pokrytí procesů týkajících se skladu kulatiny, skladu dřevní hmoty a pořezu kulatiny. Firma nyní do určité značně omezené míry pokrývá informačním systémem pouze sklad kulatiny a pořez kulatiny. Toto pokrytí je realizováno pomocí aplikace MS Excel – firma využívá vlastní řešení, které bude detailně představeno a rozebráno v kapitole 3.9.

Předmětem této práce je zlepšení pokrytí výše zmíněných procesů týkajících se skladu a pořezu kulatiny. V dalších kapitolách proto bude proveden bližší popis procesů *Příjem surové hmoty, manipulace, třídění v rámci skladu kulatiny* a procesu *Zpracování jednotlivých kusů kulatiny na výrobní lince.* Tyto procesy je možné pokrýt jedním informačním systémem – v současnosti se tak děje (viz předchozí odstavec). V další kapitole, v rámci popisu procesu *Příjem surové hmoty, manipulace, třídění v rámci skladu kulatiny* jsem tento proces částečně spojil také s procesem *Kontrola stavu skladových zásob surové hmoty*, jelikož tyto dva procesy jsou spolu úzce provázané.

3.7 Příjem surové hmoty, manipulace, třídění v rámci skladu kulatiny

Tento proces je vstupním bodem celého výrobního procesu firmy S T A R P s.r.o. Smyslem toho procesu je přijmout dodávku surové hmoty – kulatiny, zpracovat ji (tedy manipulace hmoty), zaevidovat manipulovanou hmotu a roztřídit ji ve skladu kulatiny na příslušná místa. Tento proces navazuje na proces *Kontrola stavu skladových zásob surové hmoty*, v rámci kterého probíhá i objednávka dodávek kulatiny. V této kapitole budou výše uvedené procesy blíže popsány pomocí slovního popisu, RACI matice a EPC diagramu.

3.7.1 Slovní popis

V procesu *Příjem surové hmoty, manipulace, třídění v rámci skladu kulatiny* vystupují celkem tři procesní role: *dodavatel*, *ředitel výroby* a *operátor skladu*. Dodavatel je kontaktován s požadavkem na dodání fůry kulatiny, kterou po upřesnění termínu doveze do areálu firmy. Operátor skladu před dodáním kulatiny připraví skladové místo, na které bude dodávka po doručen složena. Po složen dodávky kulatiny operátor výroby zahájí manipulaci: jednotlivé kusy kulatiny pomocí motorové pily rozřeže na kusy o délkách 3, 4 a 5 metrů. Každý takový kus pak změří – klíčová hodnota je tloušťka v čepu (tloušťka ve slabším konci měřeného kusu), označí pomocí křídy a zaznačí do sešitu, ve kterém se vedou informace o dodávkách. Poté operátor kusy kulatiny přemístí na dané místo ve skladu, na kterém je skladována kulatina dané délky a čepového průměru. Na konci směny předá sešit s informacemi o denní manipulaci kulatiny řediteli výroby, který zanese informace o manipulované kulatině – tedy změnách ve skladu – do informačního systému.

3.7.2 RACI matice

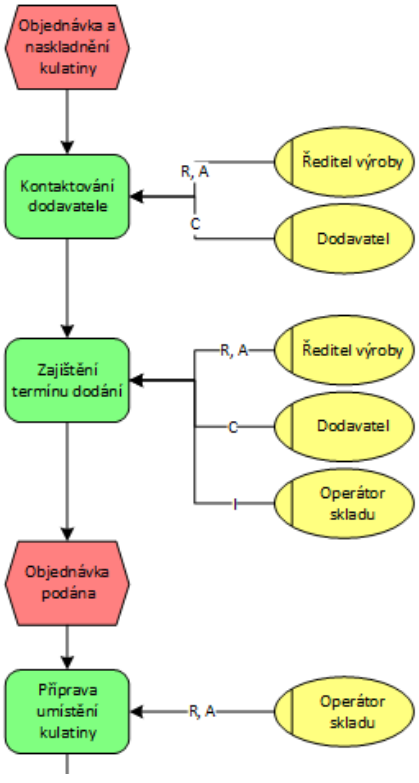
Podle slovního popisu procesu v předchozí části vzniká následující RACI matice dokumentující činnosti procesu a procesní role a rozdělení odpovědnosti za činnosti mezi procesní role.

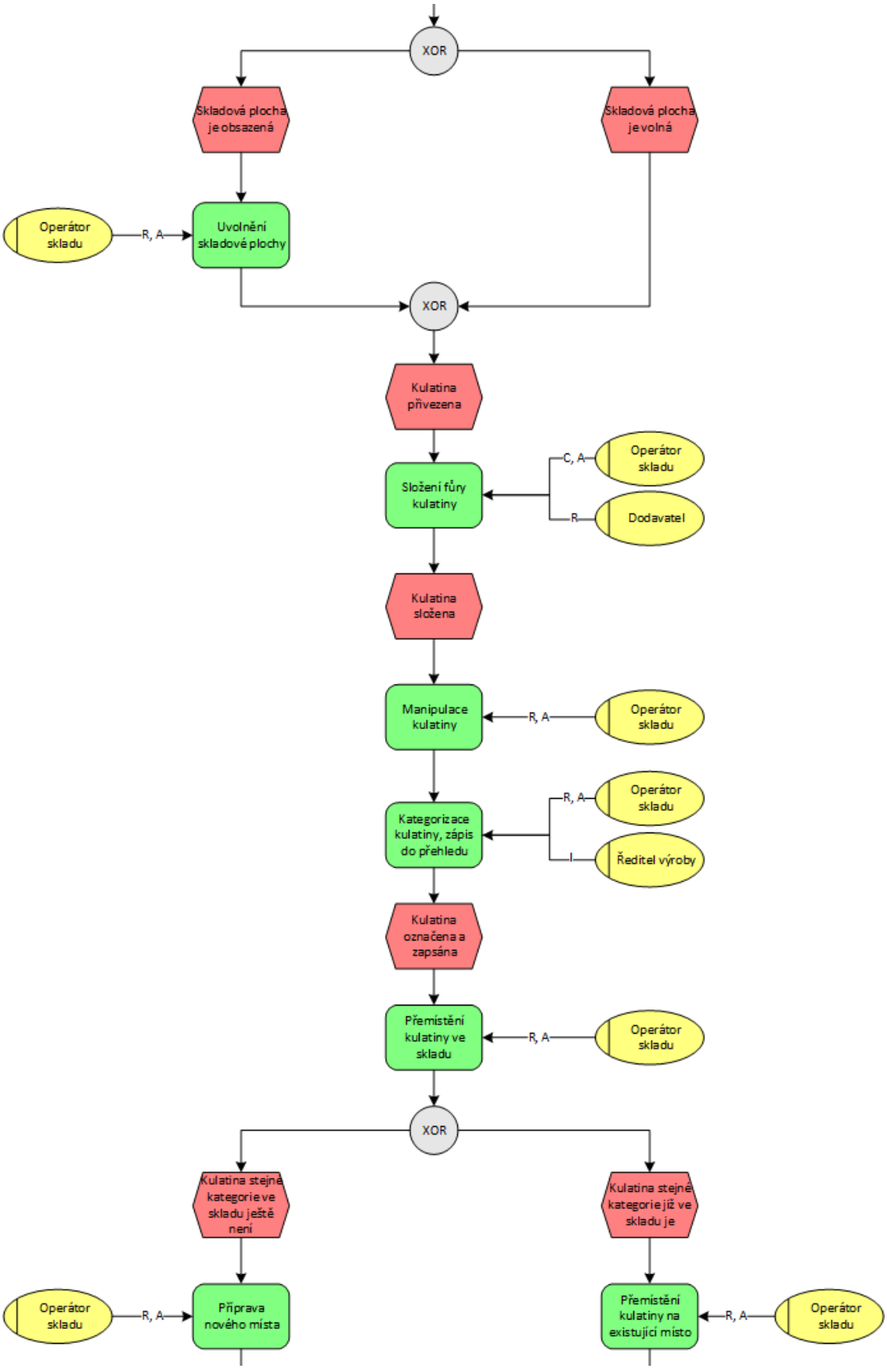
Tabulka 3 - RACI matice – Příjem surové hmoty, manipulace, třídění v rámci skladu kulatiny
(Zdroj: Vlastní zpracování)

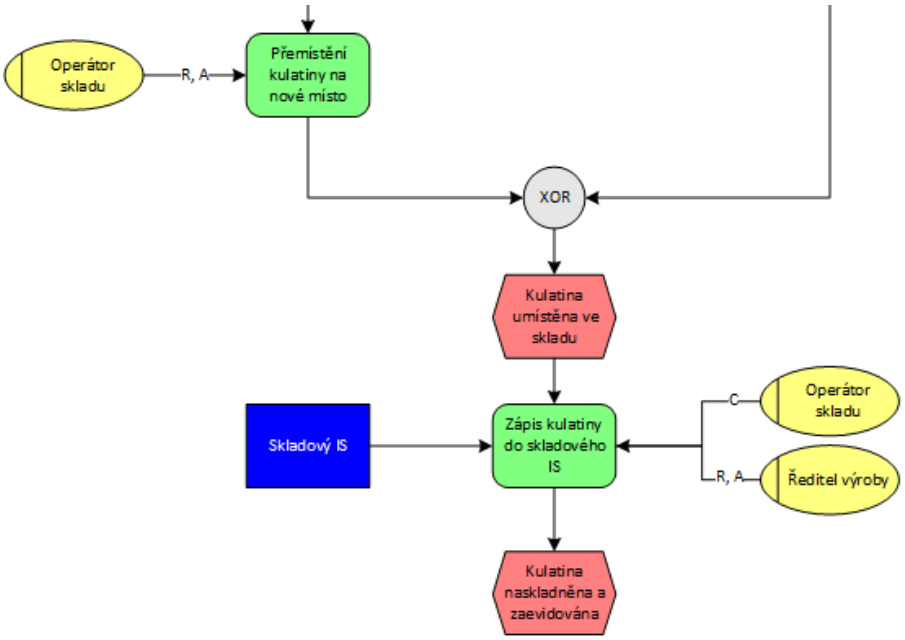
RACI Matice		Procesní role		
		Dodavatel	Ředitel výroby	Operátor skladu
Popis aktivity	Kontaktování dodavatele	C	R A	
	Zajištění termínu dodání	C	R A	I
	Příprava umístění kulatiny			R A
	Uvolnění skladové plochy			R A
	Složení fůry kulatiny	R		C A
	Manipulace kulatiny			R A
	Kategorizace kulatiny, zápis do přehledu		I	R A
	Přemístění kulatiny na existující místo			R A
	Příprava nového místa			R A
	Přemístění kulatiny na nové místo			R A
	Zápis kulatiny do skladového IS		R A	C

3.7.3 EPC diagram

V této části je proces popsán pomocí grafického nástroje EPC diagram.




XOR
Skladová plocha je obsazená
Skladová plocha je volná
Operátor skladu
R, A
Uvolnění skladové plochy
XOR
Kulatina přivezena
Složení fůry kulatiny
C, A
Operátor skladu
R
Dodavatel
Kulatina složena
Manipulace kulatiny
R, A
Operátor skladu
Kategorizace kulatiny, zápis do přehledu
R, A
Operátor skladu
I
Ředitel výroby
Kulatina označena a zapsána
Přemístění kulatiny ve skladu
R, A
Operátor skladu
XOR
Kulatina stejné kategorie ve skladu ještě není
Kulatina stejné kategorie již ve skladu je
Operátor skladu
R, A
Příprava nového místa
Přemístění kulatiny na existující místo
R, A
Operátor skladu



Obrázek 3 - EPC – Příjem surové hmoty, manipulace, třídění v rámci skladu kulatiny
(Zdroj: Vlastní zpracování)

Z diagramu je patrné, že v současnosti informační systém pokrývá pouze činnost *Zápis kulatiny do skladového IS*, kde ředitel výroby zapisuje přírůstky do skladu kulatiny do informačního systému – tedy do sešitu aplikace MS Excel (viz kapitola 3.6.2).

3.8 Zpracování jednotlivých kusů kulatiny na výrobní lince

Tento proces navazuje na proces *Zásobování výrobní linky kulatinou*. Smyslem tohoto procesu je jednotlivé kusy kulatiny, které jsou do výrobní haly přepravovány ze skladu kulatiny, zpracovat na finální produkty (latě, fošny, hranolky). Tento proces je rozsáhlý, protože každý kus kulatiny prochází na výrobní lince několika stroji a v průběhu procesu dochází i ke větvení operací se surovinami (středová část kulatiny se zpracovává jinak než boční části). Pro účely této práce však není nutný popis celého procesu včetně všech možností větvení, stačí pouze prvních několik činností, které souvisí s děním ve skladu kulatiny. Proto bude tento proces v následujících podkapitolách popsán pouze do bodu, kdy je kus kulatiny zpracován v rámové pile.

3.8.1 Slovní popis

V procesu *Zpracování jednotlivých kusů kulatiny na výrobní lince* vystupují celkem tři procesní role: *operátor skladu*, *operátor výrobní linky* a *ředitel výroby*.

Operátor skladu je zodpovědný za zásobování podávacího přístroje kulatinou podle předepsaného pořezového plánu. Podávací přístroj slouží jako spojovací článek mezi skladem kulatiny a výrobní linkou. Prvním stanovištěm výrobní linky je rámová pila (katr). Operátor výroby sedící v pojízdném vozíku – podavači kulatiny do rámové pily – nejprve naloží kus kulatiny na vozík. Poté, pokud má kus kulatiny nepravidelný tvar (např. vystouplé suky po ořezaných větvích, nepravidelný silnější konec a jiné), musí operátor výrobní linky kus kulatiny otočit tak, aby zajistil co nejhladší vstup a průchod rámovou pilou. Předtím, než kus kulatiny začne posouvat do rámové pily zaznačí do sešitu, ve kterém se vedou záznamy o pořezu během směny, právě řezaný kus (délka, čepový průměr). Poté kus kulatiny začne posouvat do rámové pily, dokud si jej nepřevezme automatický podavač v pile. Dále pak již středová část kusu kulatiny pokračuje do rozmítací pily a boční části buďto do omítací pily, nebo do sekačky. To již však není podstatné pro účely této práce, a proto již další průběh procesu nebude rozebírán. Na konci směny předá operátor výroby sešit s informacemi o denním pořezu kulatiny řediteli výroby, který zanese informace o pořezané kulatině – tedy změnách ve skladu – do informačního systému.

3.8.2 RACI matice

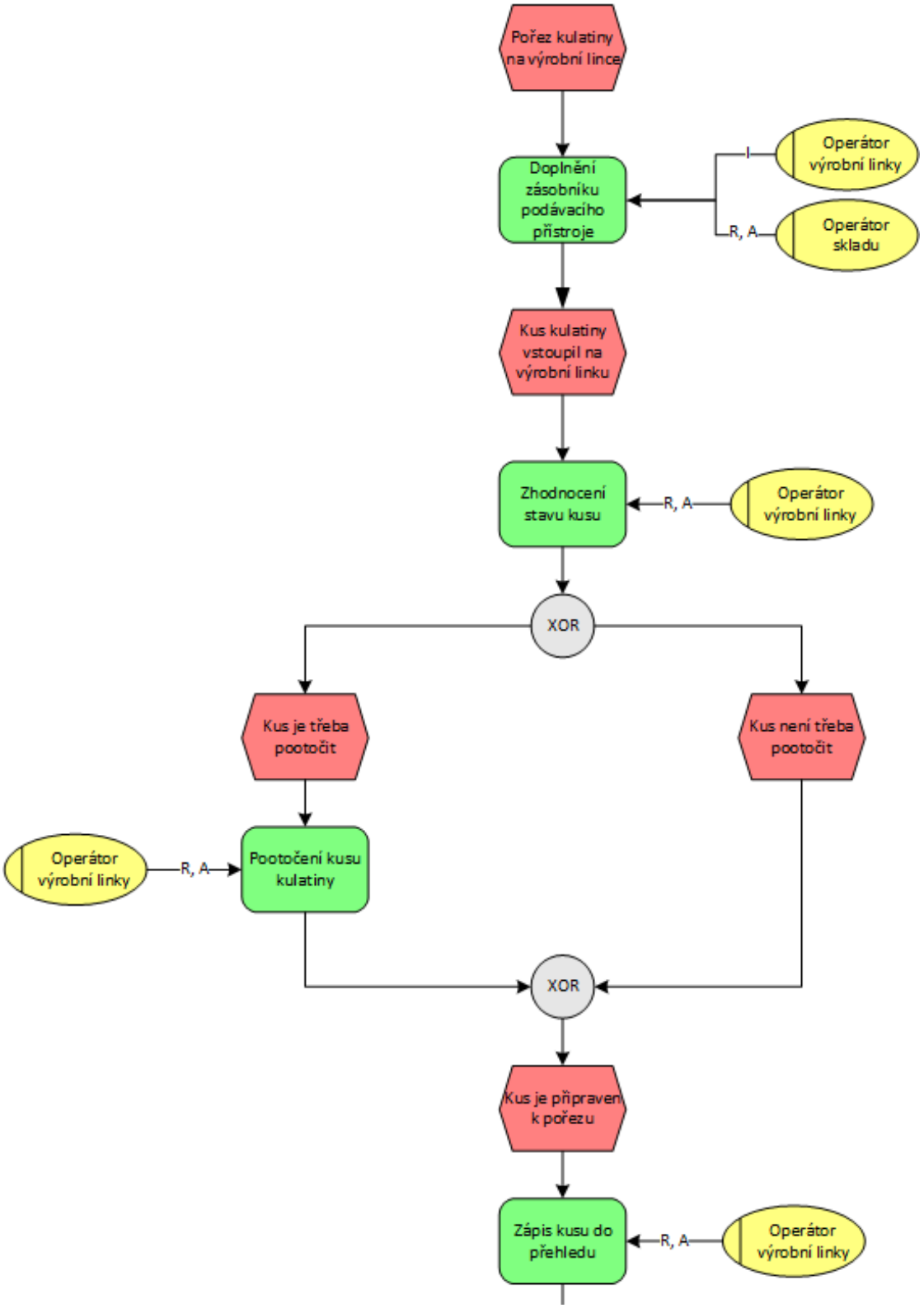
Na základě slovního popisu bude proces nyní popsán RACI maticí dokumentující činnosti procesu a procesní role a rozdělení odpovědnosti za činnosti mezi jednotlivé procesní role.

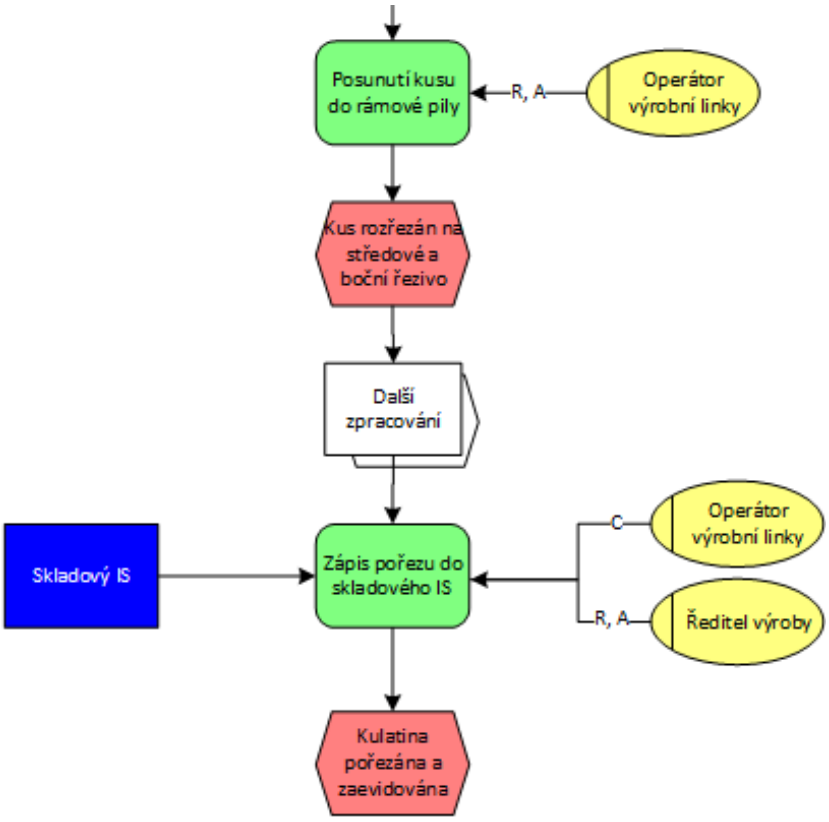
Tabulka 4 - RACI matice – Zpracování jednotlivých kusů kulatiny na výrobní lince
(Zdroj: Vlastní zpracování)

RACI Matice		Procesní role		
		Operátor výrobní linky	Ředitel výroby	Operátor skladu
Popis aktivity	Doplnění zásobníku podávacího přístroje	I		R A
	Zhodnocení stavu kusu	R A		
	Pootočení kusu kulatiny	R A		
	Zápis kusu do přehledu	R A		
	Posunutí kusu do rámové pily	R A		
	Zápis pořezu do skladového IS	C	R A	

3.8.3 EPC diagram

V této části je proces popsán pomocí grafického nástroje EPC diagram.





Obrázek 4 - EPC – Příjem surové hmoty, manipulace, třídění v rámci skladu kulatiny
(Zdroj: Vlastní zpracování)

3.9 Současný skladový IS

V této kapitole bude rozebráno současné řešení skladového informačního systému a jeho použití ve firmě S T A R P s.r.o. Nejprve bude systém popsán slovně a graficky, a poté bude provedena analýza pomocí metody HOS 8.

3.9.1 Popis současného IS

Firma v současné době používá pro účely evidence skladu a pohybů ve skladu vlastní řešení ve formě aplikace vytvořené pomocí kombinace sešitů v software MS Excel. Toto řešení vytvářel pracovník firmy, jedná se tedy o vlastní řešení problematiky vyvinuté uvnitř firmy.

Struktura systému

Informační systém je složen ze tří sešitů MS Excel, které jsou vzájemně provázané. Další, neméně důležitou součástí, jsou klasické papírové sešity, do kterých

jsou zapisovány údaje o manipulaci a pořezu kulatiny. Existuje jeden sešit pro každý proces (viz kapitoly 3.7 a 3.8). Struktura systému je tedy následující:

- Softwarová část
  - Sešit MS Excel: *manip.xls*
  - Sešit MS Excel: *pořez.xls*
  - Sešit MS Excel: *čepy.xls*
- Fyzická část
  - Sešit se záznamy o manipulaci kulatiny
  - Sešit se záznamy o pořezu kulatiny

Části softwarové jsou navzájem propojeny – vstupem pro data o příjmu a manipulaci kulatiny, tedy přírůstky skladových zásob, je soubor *manip.xls*. Soubor *pořez.xls* slouží jako vstup pro data o pořezané kulatině, tedy o úbytcích na skladě. Soubor *čepy.xls* pak propojuje předchozí soubory a agreguje data. Celý systém pak funguje na základě vzorců, které na základě vložených údajů počítají výstupy, které systém poskytuje:

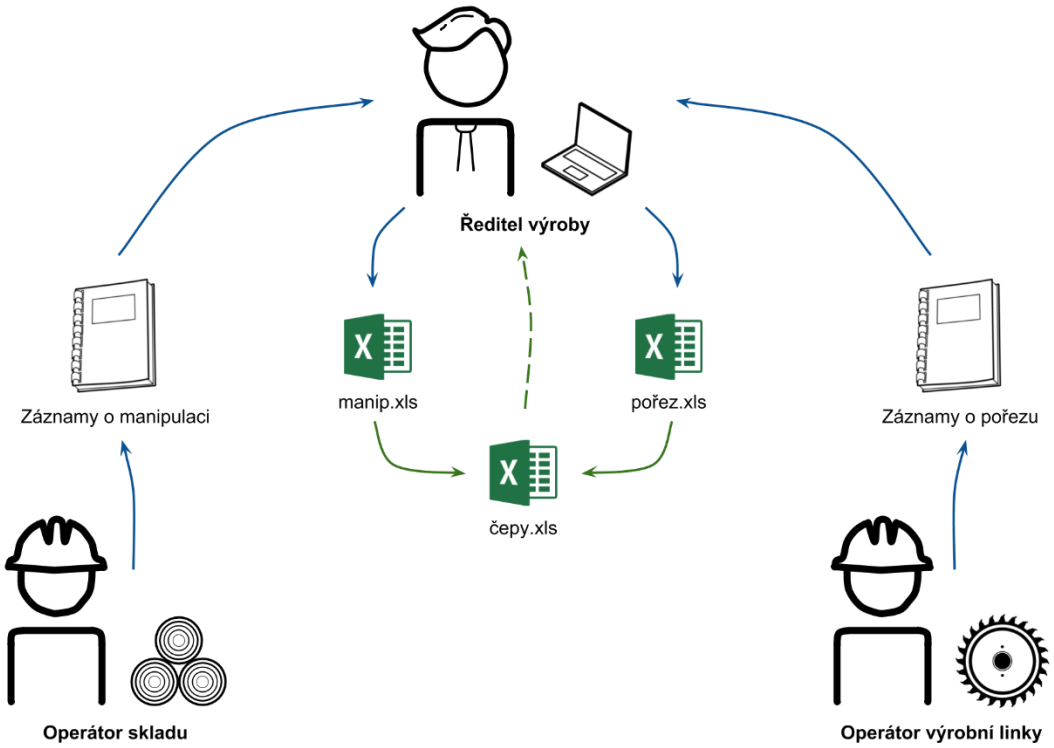
- Současný stav skladu – množství jednotlivých rozměrů kulatiny (dle délky a čepového průměru)
- Jednotlivé dodávky kulatiny za poslední měsíc
- Pořez kulatiny za jednotlivé směny za poslední měsíc včetně výkonnosti a procentuálního vyjádření plnění normy

Používání systému

Jak již bylo zmíněno v předchozích kapitolách této práce, systém je využíván v zásadě třemi zaměstnaneckými rolemi v podniku – operátorem skladu, operátorem výrobní linky a ředitelem výroby. Operátor skladu a operátor výrobní linky využívají fyzickou část systému – každý má svůj sešit, do kterého zaznamenává údaje o manipulaci, potažmo pořezu. Na konci směny oba sešity předají řediteli výroby, který data sesbíraná za danou směnu přenese do softwarové části systému. Ten postupuje tak, že sčítá kusy manipulované, případně pořezané kulatiny dané délky a čepového průměru a počet kusů daného rozměru zapisuje do příslušných buněk sešitů MS Excel *manip.xls* nebo *pořez.xls*. V souboru *čepy.xls* pak kontroluje stav skladu. Usoudí-li, že některý rozměr kulatiny není

na skladě v dostatečném počtu, může poté provést na základě svého úsudku objednávku dodávky kulatiny.

Následující obrázek graficky zobrazuje proces popsaný slovy výše. Modré šipky zobrazují tok dat ve fyzické formě (zápisy o kulatině v sešitech) a zelené šipky pak tok digitálních dat v informačním systému. Přerušovaná šipka zobrazuje zpětnou vazbu, kterou systém řediteli výroby poskytuje.



Obrázek 5 - Používání systému
(Zdroj: Vlastní zpracování)

Nevýhody současného řešení

Nejzásadnější nevýhoda současného řešení je značná uživatelská nepřívětivost a z ní pramenící pomalá práce se systémem. Zároveň, protože se jedná o MS Excel, existuje riziko poškození systému neopatrným mazáním dat, které smaže i vzorce a tím poškodí fungování systému. K tomuto scénáři již v minulosti několikrát došlo, a to zejména proto, že na konci každé měsíce je nutné současné sešity MS Excel zálohovat a vymazat data z jednotlivých dní (reprezentovaných listy sešitu). Při tomto mazání nejčastěji dojde

k chybě a zároveň je to časově náročný úkol. Potřeba mazat data o dodávkách, manipulaci a pořezu kulatiny za minulý měsíc navíc způsobuje zbytečná zdržení práce.

Z dalších nevýhod současného řešení je nutné zmínit absenci historických dat. Pro dohledání například konkrétní dodávky kulatiny z doby před několika měsíci je nutné procházet zálohované sešity, list po listu a hledat ručně. Systém také neposkytuje žádná statistická data, analýzy či grafické reprezentace, neumožňuje tvoři přehledy.

Poslední výraznou nevýhodou jsou pak průběžně vznikající nepřesnosti dat v systému. Ačkoliv data o přijaté kulatině, manipulaci a pořezu odpovídají realitě, často se stane, že datumy (reprezentované čísly listů v souborech MS Excel) neodpovídají. Tyto chyby vznikají v důsledku nevhodného návrhu systému a usnadňování si práce uživateli (např. v případě více dodávek kulatiny během jednoho dne je třeba zaznamenat data do více listů, ale pro konkrétní datum existuje pouze jeden list – uživatel pak data zapíše do vedlejšího listu, který však patří dalšímu dni). Toto, ačkoliv na souhrnný chod systému nemá výrazný vliv, značně ztěžuje zpětnou dohledatelnost záznamů.

3.9.2 SWOT analýza současného IS

V této části bude provedena SWOT analýza, ze které vzejdou poznatky o silných a slabých stránkách, příležitostech a hrozbách současného řešení. Výstupy této analýzy jsou uvedeny níže a budou později využity pro podporu rozhodnutí o dalším postupu.

Tabulka 5 - SWOT analýza
(Zdroj: Vlastní zpracování)

Silné stránky	Slabé stránky
• Dlouho zavedené řešení • Pracovníci umí systém dobře používat • Snadná modifikovatelnost	• Zastaralé řešení • Absence historických dat • Absence statistik a analýz • Nepřesnosti v datech • Snadná modifikovatelnost
Příležitosti	**Hrozby**
• Rozšíření o statistické a analytické součásti • Modifikace řešení pro zamezení nechtěných úprav • Školení uživatelů	• Nechtěné smazání aktuálních dat • Neodborný zásah do vzorců • Poškození části systému v nové verzi MS Office • Nutnost výrazného zásahu při zavedení nového rozměru kulatiny

Zhodnocení SWOT analýzy

Z provedené SWOT analýzy vzešly následující závěry: silnými stránkami současného řešení jsou zejména atributy vycházející z faktu, že systém je ve firmě využíván již mnoho let. Za tu dobu byl vyvinut do takového stavu, kdy pro běžné fungování firmy dostačuje a je pro každodenní operativní činnost plně využitelný. Pracovníci, kteří současný systém využívají, jsou s jeho funkcemi a použitím seznámeni natolik dobře, že nepotřebují žádné návody ani rady, veškerou práci zvládají v rámci možností systému efektivně a samostatně. Zároveň systém těží z faktu, že se jedná o řešení vytvořené v prostředí MS Excel, a je tedy velmi snadno upravitelné. Vyskytne-li se potřeba upravit nějaký výpočet, je celá úprava otázkou několika minut a přepsání náležitých vzorců v daných buňkách.

Slabé stránky řešení vycházejí ze skutečnosti, že systém byl vyvinut před mnoho lety, a navíc bez dostatečného návrhu. Byl vyvinut „na koleni“ a rozvíjel se podle aktuální potřeby firmy. Co však platilo před deseti a více lety již dnes neplatí – změnil se sortiment, změnily se procesy související s dodávkami kulatiny. Systém se však příliš neměnil, pouze se vždy zrovna nějak upravila ta část, která byla nejvíce dotčena. Z toho důvodu začaly v systému vznikat nepřesnosti v datech (viz předchozí kapitola). Zároveň nikdy nebyla vyvinuta žádná statistická část, která by umožňovala sledovat například nejčastěji řezaný rozměr kulatiny a další statistiky využitelné k dalšímu plánování výroby. Současně chybí analytická část, která by na základě statistických dat dokázala vytvářet vizualizace a reporty. Další nevýhodou současného řešení je také to, že je vytvořeno v prostředí MS Excel. Co je z jednoho pohledu výhoda (viz výše), to je z jiného pohledu nevýhoda – MS Excel neposkytuje rozšířené možnosti přizpůsobení uživatelského rozhraní, umožňuje nechtěnou editaci vzorců a tím poškození systému a další.

Příležitosti současného řešení skladového systému jsou zejména dodatečná tvorba již zmíněných statistických a analytických součástí, které by systému přidaly na hodnotě a využitelnosti. Také existuje možnost vyřešit některé z problémů zmíněných v předchozích částech (jako je například nechtěná editace vzorců) důsledným uzamčením příslušných buněk obsahujících vzorce. Některé uvedené problémy by také bylo možné

vyřešit pomocí školení uživatelů, které by se zaměřilo zejména na prevenci chyb vznikajících při práci se systémem a vzniku nepřesných dat.

Hrozby systému jsou reprezentovány zejména již zmíněnými neodbornými či nechtěnými zásahy do současné funkcionality systému, do vzorců a výpočtů. Určité riziko vzniká taky při každém přechodu na vyšší verzi kancelářského balíčku MS Office – může se stát, že některý vzorec či funkce využitá uvnitř současného řešení systému bude mírně či více upravena, což může poškodit či narušit současnou funkcionalitu celého systému. Poslední výraznou nevýhodou je nepřipravenost systému na nové situace ve firmě. Nastane-li například situace, kdy firma začne kupovat a zpracovávat kulatinu větších rozměrů, bude potřeba celý systém značně upravit – na mnoho míst přidat nové definice, výpočty a vzorce.

3.9.3 HOS analýza současného IS

V této části bude provedena HOS analýza současného řešení informačního systému. Analýza byla zpracována pomocí online dotazníku, který byl předložen vedení firmy. Závěry této analýzy jsou uvedeny níže a budou využity pro podporu rozhodnutí o dalším postupu.

Celkové hodnocení systému

Tabulka 6 - Analýza současného IS pomocí metody HOS8
(Zdroj: [38])

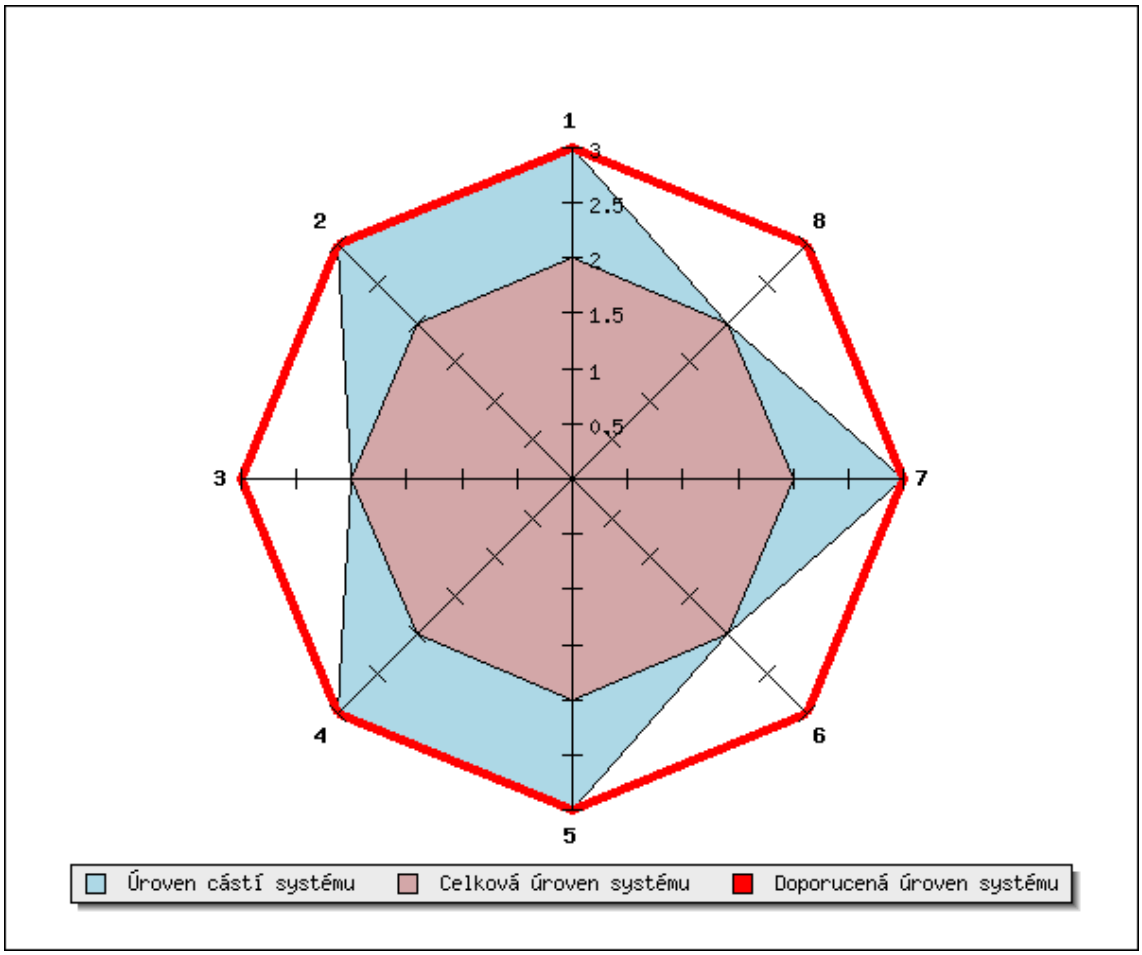
Oblast	Úroveň	
Hardware	3	Spíše dobrá
Software	3	Spíše dobrá
Orgware	2	Spíše špatná
Peopleware	3	Spíše dobrá
Dataware	3	Spíše dobrá
Zákazníci	2	Spíše špatná
Dodavatelé	3	Spíše dobrá
Management IS	2	Spíše špatná

Protože celková úroveň systému je dána jeho nejslabším článkem, a nejslabší články (orgware, zákazníci a management IS) jsou ohodnoceny stupněm 2, pak i celková úroveň systému je hodnocena jako *spíše špatná*.

[38] KOCH, Miloš. Zefis – HOS8 – posouzení vyváženosti informačního systému. Zefis.cz [online].

Grafická reprezentace hodnocení systému

V této části budou výsledky analýzy HOS8 znázorněny graficky. Grafické znázornění reprezentuje stav informačního systému pomocí osmiúhelníkového grafu, ze kterého je jasně vidět, které součásti systému nejvíce ovlivňují výsledek analýzy.



Obrázek 6 - Analýza současného IS pomocí metody HOS8
(Zdroj: [39])

Grafický výstup analýzy úrovně informačního systému potvrzuje předchozí zjištění – tedy úroveň informačního systému *spíše špatnou*. Co je důležité zmínit je fakt, že ačkoliv celkový stav je označen jako *spíše špatný*, pak doporučená hodnota, označená v grafu červenou spojnicí, je na úrovni 3, tedy *spíše dobrá*. Této úrovně dosahuje pět z celkových osmi oblastí, ostatní tři, které jsou na úrovni 2, tedy *spíše špatné*, jsou pouze o jeden stupeň hodnocení horší. Jako doporučená je zvolena úroveň 3 z toho důvodu, že

[39] KOCH, Miloš. Zefis – HOS8 – posouzení vyváženosti informačního systému. Zefis.cz [online].

v dotazníku bylo zodpovězeno na otázku důležitosti systému, že informační systém je pro činnost firmy ne nezbytně nutný, nicméně bez něj by byla práce značně zkomplikována.

Závěr hodnocení

Celková úroveň systému podle výsledku analýzy provedené metodou HOS8 je *spíše špatná*. Doporučená úroveň (kterou je však nutno chápat jako minimální požadovanou úroveň) pro daný informační systém a firemní prostředí je *spíše dobrá*. Firma tedy *nedosahuje* doporučené úrovně informačního systému. Firma by se měla zaměřit na slabé oblasti informačního systému, které jsou: *orgware*, *zákazníci* a *management IS*.

3.10 Možnosti zlepšení situace

V závěru této části práce budou představeny možnosti dalšího postupu v otázce obměny současného informačního systému. První možností je rozšíření současného řešení informačního systému, druhou možností pak je pořízení úplně nového systému.

3.10.1 Rozšíření současného IS

První možností obměny současného řešení informačního systému je rozšíření stávajícího produktu. Současné řešení by muselo být rozšířeno o uchovávání a reprezentaci statistických dat, vytvoření sešitů obsahujících grafické znázornění dat a analýz a musela by být provedena revize současné funkcionality, opraveny chyby a uzamčeny vzorce. Tímto řešením by se však systém pouze dostal do o něco lepšího a použitelnějšího stavu, ale značná část nevýhod a rizik by zůstala přítomna. Firma S T A R P s.r.o. se proto rozhodla, že současný systém již dále rozšiřovat nebude a pořídí informační systém nový.

3.10.2 Pořízení nového IS

Firma se rozhodla pořídit nový informační systém, který se bude starat o evidenci skladu kulatiny, manipulaci a pořez. Firma má tedy dvě možnosti: zakoupit již hotový skladový systém, nebo si nechat vyvinout systém na míru. Proces pořízení nového systému zakoupením existujícího řešení je nelehká, časově a finančně náročná úloha. Aby byla koupě efektivní, měla by firma věnovat čas vypracování projektu, uskutečnění výběrového řízení, výběru kandidáta, implementování systému a další věci, které celý proces obnáší. To vše je časově a finančně náročné.

Jelikož se firmě naskytla možnost nechat si nový skladový systém zhotovit v rámci této diplomové práce, a to pouze s vynaložením časových nákladů na konzultace, rozhodla se pro tuto variantu. Dostane tak systém, který bude vyhotoven na míru požadavkům firmy a uzpůsoben pro co nejsnazší a nejefektivnější použití v rámci agend příjmu a manipulace kulatiny a pořezu kulatiny, včetně veškeré funkcionality, která v současném řešení chybí.

3.10.3 Požadavky podniku na nový IS

Firma S T A R P s.r.o. si v průběhu několika konzultací, kdy byla rozebrána funkcionalita současného systému, jeho chyby a nedostatky stanovila několik základních požadavků na nový skladový systém:

- Veškerá funkcionalita současného systému
  - Evidence příjmu a manipulace kulatiny
  - Evidence pořezu kulatiny
  - Souhrny současné skladové situace
- Vytvoření způsobu zadávání údajů pro konkrétní datum – eliminace vzniku současných nepřesností
- Možnost současné práce více uživatelů
- Uchovávání historických dat a možnost jejich procházení
- Možnost definice kategorií (rozměrů) kulatiny pro případ rozšíření sortimentu
- Vytvoření statistické a analytické části – grafické zobrazení agregovaných dat
- Zpřehlednění uživatelského rozhraní

3.11 Závěr analýz

V této kapitole, analýze současného stavu, byla představena firma S T A R P s.r.o., její sortiment nabízených výrobků a služeb a organizační struktura. Poté byly popsány informační technologie, které firma využívá ke své práci. V další části byly rozebrány procesy firmy a z nich vybrány a blíže popsány dva procesy, jejichž pokrytí informačním systémem je cílem této práce. Poté bylo představeno současné řešení informačního systému, který vybrané procesy obstarává, popsáno jeho každodenní využití a jeho

nedostatky. Poté byly zpracovány analýzy SWOT a HOS8 pro současné řešení, jejichž výstupy posloužily jako podpora pro rozhodnutí o pořízení nového informačního systému. Z provedencýh analýz vyplynulo, že současný informační systém je pro potřeby firmy nedostatečný a nesplňuje požadavky firmy. Proto bylo rozhodnuto o pořízení nového informačního systému, jež bude navržen a popsán v následující kapitole, a to na základě výše popsaných požadavků firmy.

4 Vlastní návrh řešení

V této kapitole se zaměřím na detailní dekompozici požadavků na nový informační systém, tedy na určení a popsání jeho požadované funkcionality a návrh vhodného řešení realizace informačního systému. Zaměřím se také na způsob provozování systému a na monitorování provozu a událostí proběhlých v informačním systému. Dále navrhnu postup implementace informačního systému, a nakonec provedu ekonomické zhodnocení celého řešení.

4.1 Dekompozice požadavků na informační systém

Tato podkapitola se zabývá dekompozicí požadavků na nový informační systém, jež vzešly z analýz provedených v kapitole 3. Budou zde rozebrány definované požadavky na informační systém a vyvozeny postupy, kterými bude dosaženo naplnění těchto požadavků.

4.1.1 Požadavky na funkčnost

Z požadavků společnosti uvedených v kapitole 3.10.3 vyplývá, že systém bude muset být schopen provádět operace čtení, zápisu a případně mazání dat týkajících se agendy evidence příjmu a pořezu kulatiny. Následující výčet definuje požadované funkce systému a blíže specifikuje operace prováděné v rámci každé oblasti.

- Evidence příjmu a manipulace kulatiny
    - Správa dodavatelů a dopravců
        - Zápis, editace, mazání
    - Zobrazení seznamu zaznamenaných přejímek
        - Zobrazení, filtrování, mazání
    - Práce s jednotlivou přejímkou
        - Zápis, editace
- Evidence pořezu kulatiny
    - Zobrazení seznamu zaznamenaných pořezů
        - Zobrazení, filtrování, mazání
    - Práce s jednotlivým pořezem
        - Zápis, editace

- Souhrny skladové situace
    - Zobrazení celkového stavu skladu
    - Zobrazení pohybů ve skladu dle kategorie kulatiny
        - Zobrazení, filtrování
- Statistická a analytická část
    - Statistiky skladu
    - Statistiky příjmů
    - Statistiky pořezů
        - Zobrazení, filtrování
- Správa uživatele
    - Správa uživatelského účtu
        - Zobrazení, editace

4.1.2 Požadavky používání

Tato podkapitola definuje požadavky na uživatelskou stránku aplikace. Firma cílí na uživatelsky přívětivé prostředí a kvalitní UX (user experience). Systém proto musí splňovat následující požadavky.

- Snadná orientace v prostředí systému
- Jasná nabídka agend
- Přehledný způsob zadávání maticových dat do tabulek
- Aktualizace průběžných stavů dat v reálném čase
- Snadná čitelnost údajů
- Vizuálně přívětivé zobrazení statistických údajů

4.1.3 Požadavky technologické

Technologické požadavky vychází z předchozích dvou skupin požadavků. Většina technologických požadavků nebyla definována firmou, jelikož vedení firmy nemá v této oblasti potřebný přehled, a proto byly stanoveny s ohledem na prostředí ve firmě, finanční možnosti a požadavky na funkcionalitu. Jedná se o následující sadu požadavků.

- Možnost práce více uživatelů současně
- Nízké nároky na výpočetní výkon serverového stroje

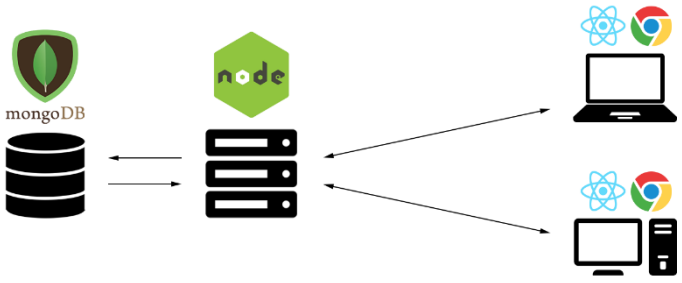
- Nenáročné a rychlé serverové aplikační prostředí
- Technologie klienta umožňující práci s výpočty v reálném čase
- Databázový stroj umožňující ukládání velkých objemů dat a provádění efektivních agregačních operací nad velkými objemy dat

4.1.4 Závěr dekompozice požadavků

Tato kapitola rozebrala požadavky na nový systém definované firmou a stanovila základ pro návrh samotné architektury aplikace a jejích jednotlivých součástí. Na základě poznatků z této kapitoly budou dále navrženy a vyvinuty konkrétní komponenty systému, jež budou obstarávat požadovanou funkcionalitu.

4.2 Architektura systému

Jelikož jedním z důležitých požadavků na nový informační systém je možnost práce více uživatelů současně, bylo rozhodnuto, že systém bude budován podle architektury klient – server. Toto řešení přináší kromě již zmíněného také další výhody, jako například možnost vývoje mobilní aplikace pro pracovníky v terénu (viz kapitola Budoucí rozšíření). Jako server bude sloužit aplikace napsaná v prostředí Node.js, jež bude spolupracovat s databázovým strojem MongoDB. Klient pak bude Single Page Aplikace běžící v prostředí webového prohlížeče, postavená na základě knihovny pro tvorbu uživatelských prostředí React. Jedná se tedy o architekturu, kde operace s daty obstarává server a uživatel pracuje s tenkým klientem. Následující obrázek graficky znázorňuje popsanou architekturu.



Obrázek 7 - Vizualizace architektury aplikace
(Zdroj: Vlastní zpracování)

Komunikační protokol

Jelikož jak serverové prostředí Node.js, tak i implementace tenkého klienta běžícího v prostředí webového prohlížeče jsou webově orientované technologie, je nasnadě zvolit jako komunikační protokol zavedený a léty ověřený protokol HTTP(S). Pro komunikaci mezi serverem a klientem tedy slouží HTTP požadavky (ty mohou nést v těle požadavku nějaká data), jež klientská aplikace odesílá na server a server vrací klientovi zpět odpovědi na základě jeho požadavků. Pro přístup k serveru publikuje server sadu zdrojů a operací se zdroji, jež mohou klienti využívat. Tento přístup servírování zdrojů je v souladu s filosofií REST rozhraní (více v kapitole 4.3.3).

4.3 Server

Tato kapitola se zabývá popisem serverové části systému. Budou zde popsány technologie, se kterými server pracuje, definovány postupy komunikace s klienty a rozebrány technologie a postupy práce s úložištěm dat.

4.3.1 Technologie využité v serverové části

Jak již bylo zmíněno v předchozích kapitolách, server je založen na prostředí Node.js. Toto prostředí bylo zvoleno z důvodu jeho snadné implementace, nízkých nároků na výpočetní výkon, snadnou obsluhu a rychlý proces vývoje funkcionality. Dalším důvodem je, že Node.js byl vytvořen jako ideální nástroj pro implementaci REST rozhraní, jež je využito v rámci vytvářeného systému a tvoří vstupní bod pro klienty pro komunikaci se serverem.

Moduly Node.js

Samotný Node.js je velmi schopný a všestranný nástroj, tu pravou sílu však získává díky nespočetnému množství rozšíření a doplňků, jež značným způsobem zjednodušují práci. V rámci vytvářeného systému byla využity zejména dvě rozšíření: *restify* a *mongoose*. První jmenovaný zjednodušuje, definici a implementaci REST rozhraní v rámci Node.js a druhý jmenovaný slouží ke komunikaci s databázovým strojem – umožňuje definovat modelová schémata a provádět operace s MongoDB databází (viz dále v části Databázový stroj). Nakonec je třeba zmínit důležitou součást serverové části aplikace, a to rozšíření zajišťující autentizaci a autorizaci uživatelů vůči poskytovaným zdrojům. Tato agenda je pokryta rozšířením *oauth2-server*, které jak již

název napovídá, implementuje v rámci serverové části aplikace autorizační a autentizační protokol Oauth 2.0 (více v kapitole 4.3.4). Následující tabulka obsahuje výčet použitých rozšíření (modulů) v konkrétní implementaci Node.js v rámci serverové části vyvíjené aplikace.

Tabulka 7 - Seznam použitých modulů pro Node.js
(Zdroj: vlastní zpracování)

Modul	Použití
bcrypt	Šifrování uživatelských hesel
body-parser	Zpracování přijatých dat v těle HTTP požadavku
moment	Práce s datumy
mongoose	Komunikace s MongoDB databází
oauth2-server	Autorizační a autentizační rozhraní
restify	Publikace REST zdrojů a metod

Databázový stroj

Jako databázový stroj použitý pro správu dat a operace s daty byla zvolena databáze MongoDB. Jedná se o dokumentově orientovanou NoSQL databázi, jež nevyužívá tradičního relačního schématu, ale namísto toho pracuje s dokumenty, jež mají podobu velmi blízkou JSON formátu. Tato databáze byla zvolena z důvodu skvělé kompatibility s prostředím Node.js, vysoké optimalizace a dokumentově orientovanému schématu, jež lépe odpovídá požadavkům aplikace. Konkrétní struktura databáze, jednotlivé kolekce a schémata dokumentů jsou rozebrány v kapitole 4.3.2.

4.3.2 Databázová struktura

Základem stabilní a výkonné aplikace je dobře navržená databázová struktura a vhodně zvolené datové modely. Tradiční SQL databáze jsou postaveny na principu, kdy model je reprezentován tabulkou a záznam řádkem. MongoDB pracuje s kolekcemi místo tabulky a dokumenty místo řádků. Navíc MongoDB umožňuje ukládat zanořené dokumenty a vytvářet složitější a hlubší datovou strukturu. Na tuto skutečnost je třeba brát ohled a využívat ji při návrhu datového modelu. V této kapitole bude definován datový model aplikace, představeny jednotlivé kolekce a definována schémata jednotlivých kolekcí. Datový model vychází z požadavků na funkcionalitu sytému definovaných 4.1.1.

Nejdůležitějšími součástmi aplikace jsou agendy příjmu kulatiny, pořezu kulatiny a skladu. Od těchto údajů se odvíjí většina dalších funkcí systému. Nejprve tedy budou popsány potřebné datové modely pro zajištění této agendy a poté budou následovat další podpůrné dokumenty. Přestože struktura databázových záznamů (dokumentů) v prostředí MongoDB je ve své podstatě vyjádřena pomocí JSON formátu, bude z důvodu přehlednosti a typografie využit tabulkový popis modelů.

Tabulka 8 - Schéma subdokumentu WarehouseUnit
(Zdroj: Vlastní zpracování)

Subdokument WarehouseUnit		
Atribut	**Typ**	**Poznámka**
_id	ObjectId	
diameter	Number	
length	Number	
amount	Number	
volume	Number	

Tabulka 9 - Schéma dokumentu Handling
(Zdroj: Vlastní zpracování)

Dokument Handling		
Atribut	**Typ**	**Poznámka**
_id	ObjectId	
date	ISODate	
delivery_note_number	String	
supplier	ObjectId	Reference na dokument Supplier
supplier_center	String	
supplier_claimed_volume	Number	
supplier_delivery_note_number	String	
hauler	ObjectId	Reference na dokument Hauler
timber_type	String (enum)	
timber_volume_d	Number	
handling	Array	Pole subdokumentů WarehouseUnit
created_at	ISODate	
updated_at	ISODate	

Tabulka 10 - Schéma dokumentu Sawing
(Zdroj: Vlastní zpracování)

Dokument Sawing		
Atribut	**Typ**	**Poznámka**
_id	ObjectId	
date	ISODate	
performance_percent	Number	
produced_center_bad	Number	
produced_center_good	Number	
produced_shorts	Number	
produced_side	Number	
produced_percent_center_good	Number	
produced_percent_center_total	Number	
produced_percent_total	Number	
time_day	Number	
time_fault	Number	
time_change	Number	
time_air	Number	
time_other	Number	
time_net	Number	
sawing	Array	Pole subdokumentů WarehouseUnit
created_at	ISODate	
updated_at	ISODate	

Tabulka 11 - Schéma dokumentu Warehouse
(Zdroj: Vlastní zpracování)

Dokument Warehouse		
Atribut	**Typ**	**Poznámka**
_id	ObjectId	
date	ISODate	
handling	ObjectId	Reference na dokument Handling
sawing	ObjectId	Reference na dokument Sawing
warehouse	Array	Pole subdokumentů WarehouseUnit
created_at	ISODate	
updated_at	ISODate	

Následuje popis ostatních doplňujících modelů k modelům týkajícím se agendy evidence skladu. Jedná se o kolekce Supplier – tedy dodavatel, Hauler – tedy dopravce a kolekci Settings – tedy obecné nastavení aplikace.

Tabulka 12 - Schéma dokumentu Supplier
(Zdroj: Vlastní zpracování)

Dokument Supplier		
Atribut	**Typ**	**Poznámka**
_id	ObjectId	
name	String	
phone	String	
email	String	
street	String	
city	String	
zip	String	
created_at	ISODate	
updated_at	ISODate	

Tabulka 13 - Schéma dokumentu Hauler
(Zdroj: Vlastní zpracování)

Dokument Hauler		
Atribut	**Typ**	**Poznámka**
_id	ObjectId	
name	String	
phone	String	
email	String	
street	String	
city	String	
zip	String	
created_at	ISODate	
updated_at	ISODate	

Tabulka 14 - Schéma dokumentu Settings
(Zdroj: Vlastní zpracování)

Dokument Settings		
Atribut	**Typ**	**Poznámka**
_id	ObjectId	
name	String	
value	Mixed	

Poslední skupinou modelů jsou modely sloužící ke správě uživatelů, jejich autentizaci, autorizaci a kontroly přístupů. Jedná se o modely User, který drží informace o uživatelích systému a modely OauthClient a OAuthAccessToken, jež slouží rozšíření

oauth2-server pro správu informací o povolených klientských aplikacích a přístupových tokenech.

Tabulka 15 - Schéma dokumentu User
(Zdroj: Vlastní zpracování)

Dokument User		
Atribut	**Typ**	**Poznámka**
_id	ObjectId	
username	String	
password	String	
email	String	
firstname	String	
lastname	String	
scope	String	
created_at	ISODate	
updated_at	ISODate	

Tabulka 16 - Schéma dokumentu OauthClient
(Zdroj: Vlastní zpracování)

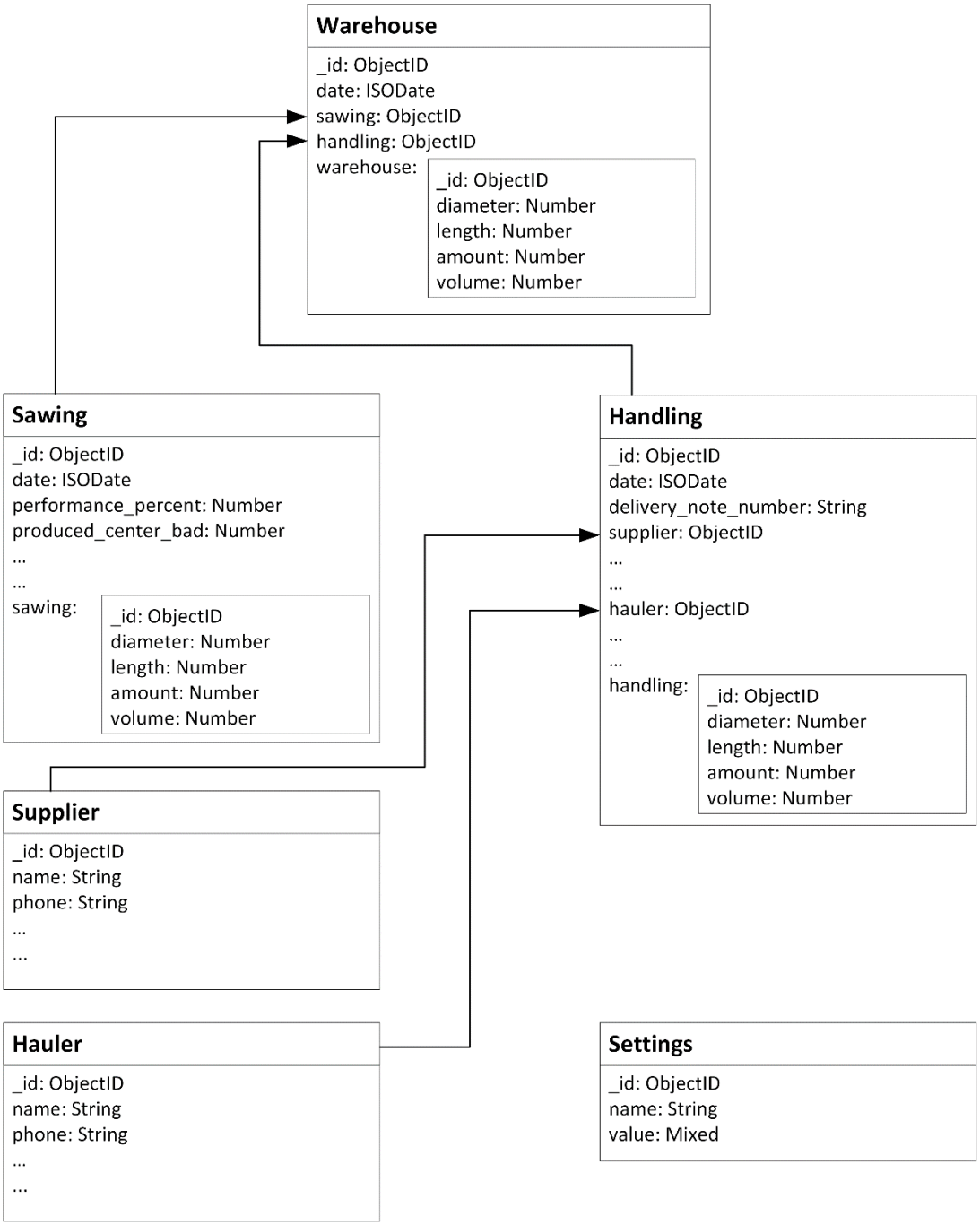
Dokument OAuthClient		
Atribut	**Typ**	**Poznámka**
_id	ObjectId	
name	String	
clientId	String	
clientSecret	String	
redirectUri	String	
grants	Array	Pole typů String
scope	String	

Tabulka 17 - Schéma dokumentu OAuthAccessToken
(Zdroj: Vlastní zpracování)

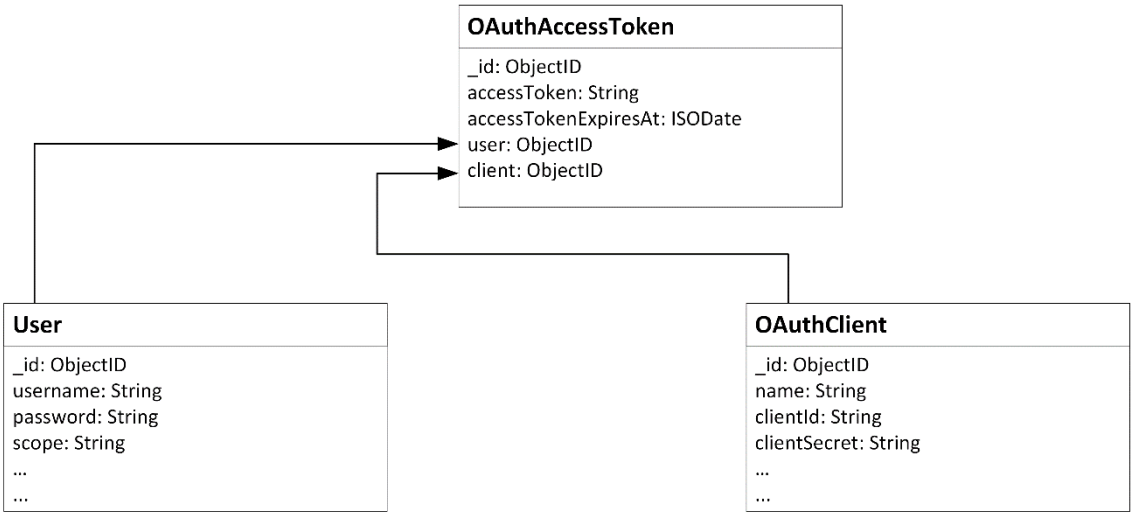
Dokument OAuthAccessToken		
Atribut	**Typ**	**Poznámka**
_id	ObjectId	
accessToken	String	
accessTokenExpiresAt	ISODate	
user	ObjectID	Reference na dokument User
client	ObjectID	Reference na dokument OAuthClient

Vztahy v databázi

Ačkoliv MongoDB není relační databáze, a proto nelze vztahy mezi jednotlivými schématy tradičním relačním (ER) diagramem, je vhodné vztahy v rámci databáze vyjádřit pomocí grafického znázornění. Toto znázornění zobrazuje vztahy mezi kolekcemi a dokumenty, jež byly popsány v předchozí části této kapitoly.



Obrázek 8 - Diagram vztahů mezi dokumenty
(Zdroj: Vlastní zpracování)



Obrázek 9 - Diagram vztahů mezi dokumenty
(Zdroj: Vlastní zpracování)

4.3.3 Zdroje a REST rozhraní

Na základě definovaných požadavků a na základě datového modelu aplikace byly definovány zdroje, k nimž klientská aplikace přistupuje a provádí s nimi CRUD operace. Vytvářený systém definuje osm zdrojů, s nimiž lze provádět různé operace. Seznam zdrojů, odpovídajících URL adres a možných metod je vypsán v následující tabulce.

Tabulka 18 - Identifikace zdrojů a operací se zdroji
(Zdroj: Vlastní zpracování)

Zdroj	URL		Akce
Uživatel	POST	/users	Vytvoření uživatele
	GET	/user	Získání přihlášeného uživatele
	GET	/users/:id	Získání uživatele podle ID
	PUT	/users/:id	Editace uživatele podle ID
Uživatelovo heslo	PUT	/users/:id/password	Změna uživatelova hesla
Dodavatel	GET	/suppliers	Získání seznamu dodavatelů
	POST	/suppliers	Vytvoření dodavatele
	GET	/suppliers/:id	Získání dodavatele podle ID
	PUT	/suppliers/:id	Editace dodavatele podle ID
	DEL	/suppliers/:id	Smazání dodavatele podle ID
Dopravce	GET	/haulers	Získání seznamu dopravců
	POST	/haulers	Vytvoření dopravce
	GET	/haulers /:id	Získání dopravce podle ID
	PUT	/haulers /:id	Editace dopravce podle ID
	DEL	/haulers /:id	Smazání dopravce podle ID
Přejímka	GET	/handlings	Získání seznamu přejímek
	POST	/handlings	Vytvoření přejímky
	GET	/handlings /:id	Získání přejímky podle ID
	PUT	/handlings /:id	Editace přejímky podle ID
	DEL	/handlings /:id	Smazání přejímky podle ID
Pořez	GET	/sawings	Získání seznamu pořezů
	POST	/sawings	Vytvoření pořezu
	GET	/sawings /:id	Získání pořezu podle ID
	PUT	/sawings /:id	Editace pořezu podle ID
	DEL	/sawings /:id	Smazání pořezu podle ID
Sklad	GET	/warehouse	Získání současného stavu skladu
	GET	/warehouse/groups	Získání skladových pohybů dle skupin
Statistiky	GET	/stats/warehouse	Získání statistik skladu
	GET	/stats/handling	Získání statistik příjmu
	GET	/stats/sawing	Získání statistik pořezu
Nastavení	GET	/settings	Získání nastavení

4.3.4 Zabezpečení, autentizace a autorizace

Zabezpečení zdrojů proti neoprávněnému nakládání je nesmírně důležitou součástí každé serverové implementace. V současnosti je víceméně standardem využívat pro zabezpečení veřejných i neveřejných API rozhraní implementujících REST architekturu autorizační protokol OAuth 2.0. Tento protokol je využit i v rámci vyvíjeného systému. Serverovou stranu komunikace (a zároveň správu klientů, uživatelů a přístupových tokenů) zajišťuje modul Node.js *oauth2-server*.

OAuth 2.0 podporuje celou řadu možností (grant types), jak získat přístupový token, jež je následně využit při operacích se zdroji. Vyvíjená aplikace implementuje možnost *Resource Owner Password Credentials*, často označovanou pouze jako *password*. Tato možnost je využívána pouze při práci s klientskými aplikacemi, kterým je možno stoprocentně důvěřovat, jelikož nakládání s uživatelovým uživatelským jménem a heslem zajišťuje právě klientská aplikace. Pro úspěšné získání přístupového tokenu se však musí nejprve autentizovat samotná klientská aplikace. To zajišťuje, že server nebude zneužit (za předpokladu zachování utajení identifikátoru a hesla klientské aplikace). Celý průběh autentizace pak probíhá následovně:

1. Uživatel zadá do přihlašovacího formuláře své uživatelské jméno a heslo
2. Klientská aplikace odešle požadavek `POST /oauth/token`
    - Včetně položky hlavičky `Authorization: Basic <base64secret>` kde `<base64secret>` je base64 zakódovaný identifikátor a heslo klientské aplikace
    - Tělo požadavku obsahuje hodnoty `grant_type, username, password` kde `grant_type=password` a `username` a `password` jsou uživatelem zadané hodnoty
3. Server nejprve dekóduje identifikaci klientské aplikace a zjistí, zda klientská aplikace existuje a poskytla správný identifikátor a heslo, a zda je oprávněna použít zvolenou metodu získání přístupového tokenu (v tomto případě *password*)
4. Poté server identifikuje uživatele podle předaného parametru `username` a ověří, zda zadal správné heslo

5. Pokud všechna ověření skončí úspěchem, vrátí server odpověď se stavovým kódem 200 a v těle odpovědi předá přístupový token; v opačném případě vrátí odpověď se stavovým kódem 401 a v těle odpovědi předá důvod neúspěšné autentizace
6. Klient uloží do lokálního úložiště přístupový token a všechny další požadavky na operace se zdroji směřující na server doplní hlavičkou `Authorization: Bearer <access token>` kde `<access token>` je obdržený kód přístupového tokenu
7. Server na základě obdrženého přístupového tokenu ověří, že uživatel byl úspěšně autentizován a že token je platný, následně dále zpracuje požadavek na operaci se zdrojem. Pakliže předchozí ověření není úspěšné, vrátí server odpověď se stavovým kódem 401 a důvodem chyby; nepokračuje ve zpracování požadavku na operaci se zdrojem

Veškerá popsaná komunikace samozřejmě spoléhá na fakt, že komunikace mezi klientskou aplikací a serverem probíhá přes zabezpečený kanál, tedy v tomto případě přes komunikační protokol HTTPS.

4.4 Klientská aplikace

Tato kapitola se zaměří na popis klientské aplikace, se kterou budou pracovníci firmy každý den pracovat. Tato aplikace komunikuje se serverem, jež byl popsán v předchozí kapitole. V této kapitole budou nejprve představeny a detailněji popsány moduly aplikace, poté bude popsán vizuální styl aplikace z hlediska uživatelského rozhraní (UI) a uživatelské zkušenosti (UX) a nakonec bude rozebrána funkcionalita aplikace a popsán způsob, jakým aplikace funguje.

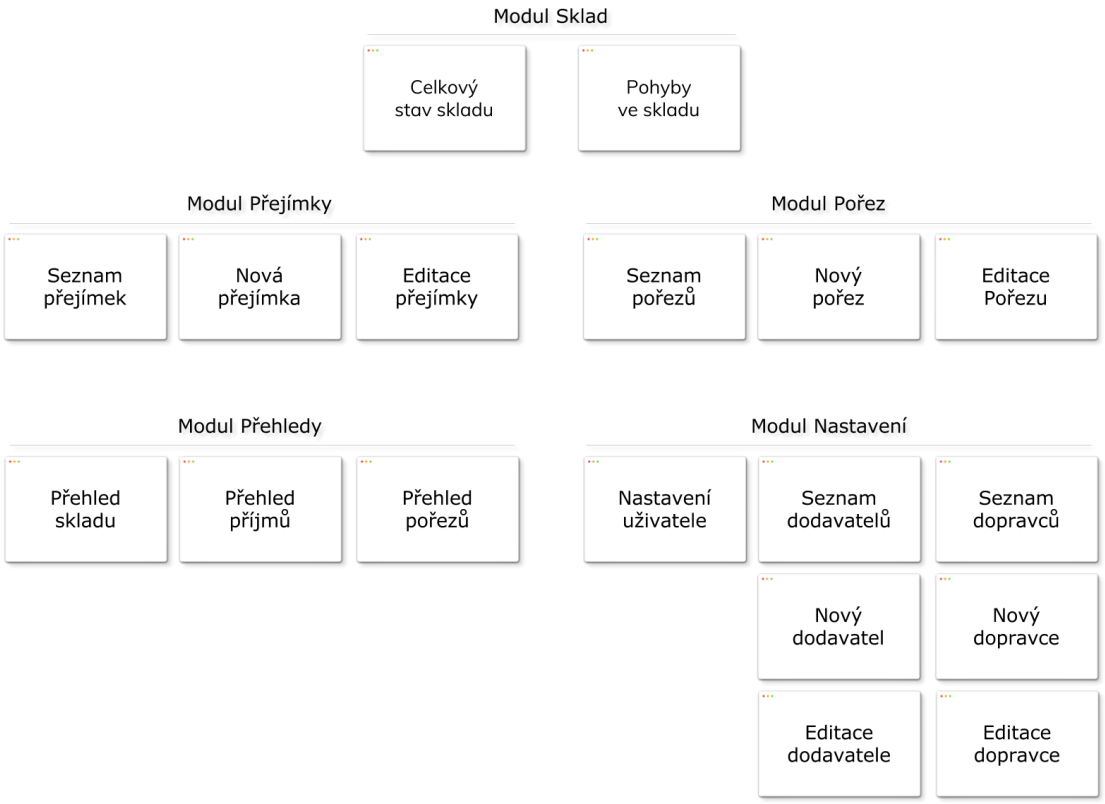
4.4.1 Moduly aplikace

Celá aplikace se skládá ze čtyř hlavních modulů plus modulu nastavení. Moduly byly navrženy tak, aby pokrývaly a byly schopny pracovat s veškerými zdroji, jež jsou k dispozici (viz kapitola 4.3.3). Seznam všech modulů se stručným popisem každého z nich je vypsán v následující tabulce. Dále v této kapitole bude každý modul rozebrán podrobněji.

Tabulka 19 - Přehled modulů klientské aplikace
(Zdroj: Vlastní zpracování)

Modul	Popis
Sklad	Výchozí modul aplikace. Agreguje data z příjmu a výdeje zboží (tedy přejímek a pořezu kulatiny) a zobrazuje celkové stavy skladu dle daných kategorií a skladové pohyby za zvolené časové období.
Přejímky	Přehled a správa přijatého (manipulovaného) zboží – kulatiny. Možnost zobrazení seznamu přejímek a filtrování v rámci seznamu. Možnost vytvoření nové přejímky a editace existujících.
Pořez	Přehled a správa vydaného (pořezaného) zboží – kulatiny. Možnost zobrazení seznamu pořezů a filtrování v rámci seznamu. Možnost vytvoření nového záznamu pořezu a editace existujících.
Přehledy	Modul statistických údajů ve formě grafických přehledů. Umožňuje sledovat aktuální stav a historický vývoj údajů skladu, příjmu zboží a pořezu za zvolená období ve formě přehledných grafů.
Nastavení	Nastavení aplikace a uživatelského účtu. Správa seznamu dodavatelů zboží a dopravců zboží. Možnost zobrazení seznamu dodavatelů a dopravců, vytvoření nového či editace stávajících.

Každý modul se skládá z jednotlivých součástí, jež jsou uživateli prezentovány jako jednotlivé obrazovky aplikace, a každá z nich slouží k specifickému účelu. Grafický přehled modulů a jejich součástí je zobrazen na schématu na další stránce, které přehledně zobrazuje strukturu aplikace.



Obrázek 10 - Přehled modulů a obrazovek systému
(Zdroj: Vlastní zpracování)

Mezi jednotlivými obrazovkami a moduly uživatel přechází pomocí bočního menu (viz dále v kapitole 4.4.2) nebo pomocí akcí v systému (typicky například uložení dokumentu či zrušení editace dokumentu). Schéma také zobrazuje určitou hierarchii systému. Nejčastěji se uživatelé budou pohybovat v modulu skladu, poté v modulech přejímek a pořezu. Modul přehledů bude využíván občasně pro účely sledování trendů za určitou obdobu a modul nastavení bude využíván nejméně. V následujících částech každý z modulů detailněji popíšu.

Modul sklad

Sklad je nejdůležitější součást celé aplikace. Uživatel (ředitel výroby) bude tento modul využívat každý den pro kontrolu množství zboží (kulatiny) na skladě, přičemž maticové zobrazení skladových zásob podle kategorií (délka a čepový průměr) přehledně ukazuje, jaký je stav skladu v každé ze sledovaných kategorií. Ředitel výroby pak podle úrovní skladových zásob rozhoduje, jaký sortiment se bude daný pracovní den zpracovávat (v návaznosti na další vstupní parametry, jako je například požadovaný

konečný výrobek). Zároveň dle toho sestavuje požadavky na další objednávku kulatiny tak, aby skladové zásoby v daných úrovních udržel v požadované hladině. Obrazovka pohybů ve skladě pak zobrazuje dle jednotlivých dnů a zvoleného čepového průměru, jaký byl daný den příjem na sklad a výdej ze skladu.

Modul přejímky

Modul přejímek obsahuje v zásadě dvě obrazovky. První obrazovka obsahuje seznam přejímek za zvolené časové období. Uživatel si může zvolit vlastní časové období a dále filtrovat přejímky podle dodavatele. Z této obrazovky je rovněž možné přejít na editaci přejímky, případně přejímku smazat. Další obrazovkou je obrazovka pro přidání nové přejímky. Zde uživatel vyplňuje základní data o přejímce (dodavatel, číslo dodacího listu atd.) a hlavně data o přijatém zboží. To je realizováno pomocí maticového formuláře dle délek a čepových průměrů. Editace přejímky pak využívá stejnou obrazovku jako přidávání nové přejímky.

Modul pořez

Modul pořez má z obecného hlediska stejnou strukturu, jako model přejímek. Opět je úvodní obrazovkou přehled existujících pořezů s možností dalšího filtrování dle zvoleného rozsahu datumů. Stejně jako u přehledu přejímek je možné z této obrazovky přejít na editaci pořezu, nebo daný pořez vymazat. Přidávání pořezu má opět obdobnou strukturu jako přidávání přejímky. Nejprve uživatel zadá základní informace o pořezu (časové údaje o směně, reálně vyprodukované množství atd.) a poté zadává do stejného formuláře jako u přejímek data o pořezaném zboží. Editace pořezů využívá stejnou obrazovku jako přidávání nového pořezu.

Modul přehledy

Modul přehledů je rozdělen do tří obrazovek podle kategorie daného přehledu. První obrazovka obsahuje přehledy skladu. To znamená tři grafické vizualizace vývoje skladových zásob za zvolené časové období (ve výchozím stavu 1 měsíc). Prvním grafem je sloupcový graf vývoje celkového stavu zásob bez ohledu na kategorii (délku či čepový průměr). Druhým grafem je skupinový sloupcový graf, který zobrazuje vývoj stavu skladu dle jednotlivých délek kulatiny. Třetí graf zobrazuje stav skladu podle jednotlivých čepových průměrů. Druhá obrazovka se zabývá přehledy příjmu zboží. Obsahuje dva grafy, kdy první z nich ukazuje množství přijatého zboží za jednotlivé dny.

Druhý graf – tentokrát koláčový – zobrazuje podíl dodávek zboží dle jednotlivých dodavatelů. Třetí a poslední obrazovka se zabývá přehledy pořezu. Opět obsahuje dva grafy. První z nich ukazuje množství pořezaného zboží za zvolené období dle jednotlivých dnů. Druhý zobrazuje vývoj plnění normy pořezu za časové období (výpočet normy pořezu je probrán v kapitole 4.4.3).

Modul nastavení

Modul nastavení obsahuje nejvíce obrazovek. První obrazovka se týká nastavení uživatele. Aktuálně přihlášený uživatel si zde může upravovat profilové informace (jméno, příjmení, kontakt) a také měnit přístupové heslo do systému. Další obrazovky se vážou ke správě dodavatelů a dopravců. Tyto dvě skupiny jsou si velmi podobné a obrazovky pro každou z nich vypadají víceméně totožně. První obrazovka vždy zahrnuje seznam dodavatelů/dopravců, odkud je možno přejí na editaci dodavatele/dopravce, případně je smazat. Vytvoření a editace dodavatele/dopravce pak spočívá ve vyplnění jednoduchého formuláře, kam uživatel zadává základní informace o dodavateli či dopravci.

4.4.2 Vizuální styl aplikace

Jedním z požadavků aplikace kladených firmou byl požadavek na přehledné uživatelské prostředí, v němž se dobře orientuje a snadno se používá. Proto bylo rozhodnuto využívat v rámci aplikace následující vizuální prvky:

- Pastelové barvy příjemné na oči
- Výrazné a jasně vizuálně definované prvky
- Jednoduché, základní tvary
- Jednoduché, jednobarevné vektorové ikony
- Větší písmo a prvky
- Inspirace Material Designem

Barevná paleta

Prvním krokem bylo zvolit barvy využité napříč systémem. Jelikož systém je orientovaný na pilařský provoz a vstupní surovina je přírodního charakteru, nabízel se nějaký odstín zelené. Navíc většina firemních materiálů (logo, vizitky atd.) zahrnují zelenou barvu, bylo rozhodnuto o použití pastelového odstínu zelené. Kombinace bílé a

pastelově zelené barvy je navíc přijatelně kontrastní, vypadá na počítačových obrazovkách dobře a je příjemná na oči.

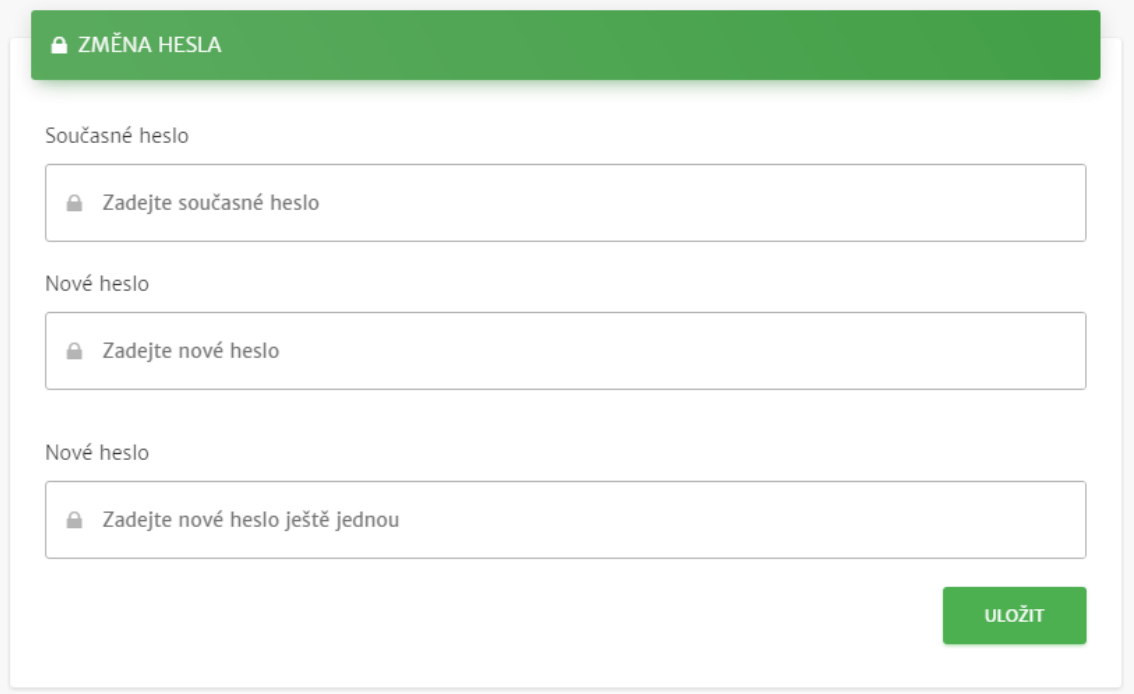
Jako primární barva tedy byla zvolena zelená barva z palety barev Material Designu nazvaná jednoduše Green (jedná se o základní odstín zelené z dané palety), jež je vyjádřena hexadecimálním kódem #4CAF50.

Material Design Green
#4CAF50, RGB (76, 175, 80)

Dále se pracuje pouze s bílou barvou využitou pro pozadí obrazovek a odstíny šedé, jež jsou využity pro barvu textu, ohraničení zadávacích polí a tabulek a ikon.

Grafické prvky

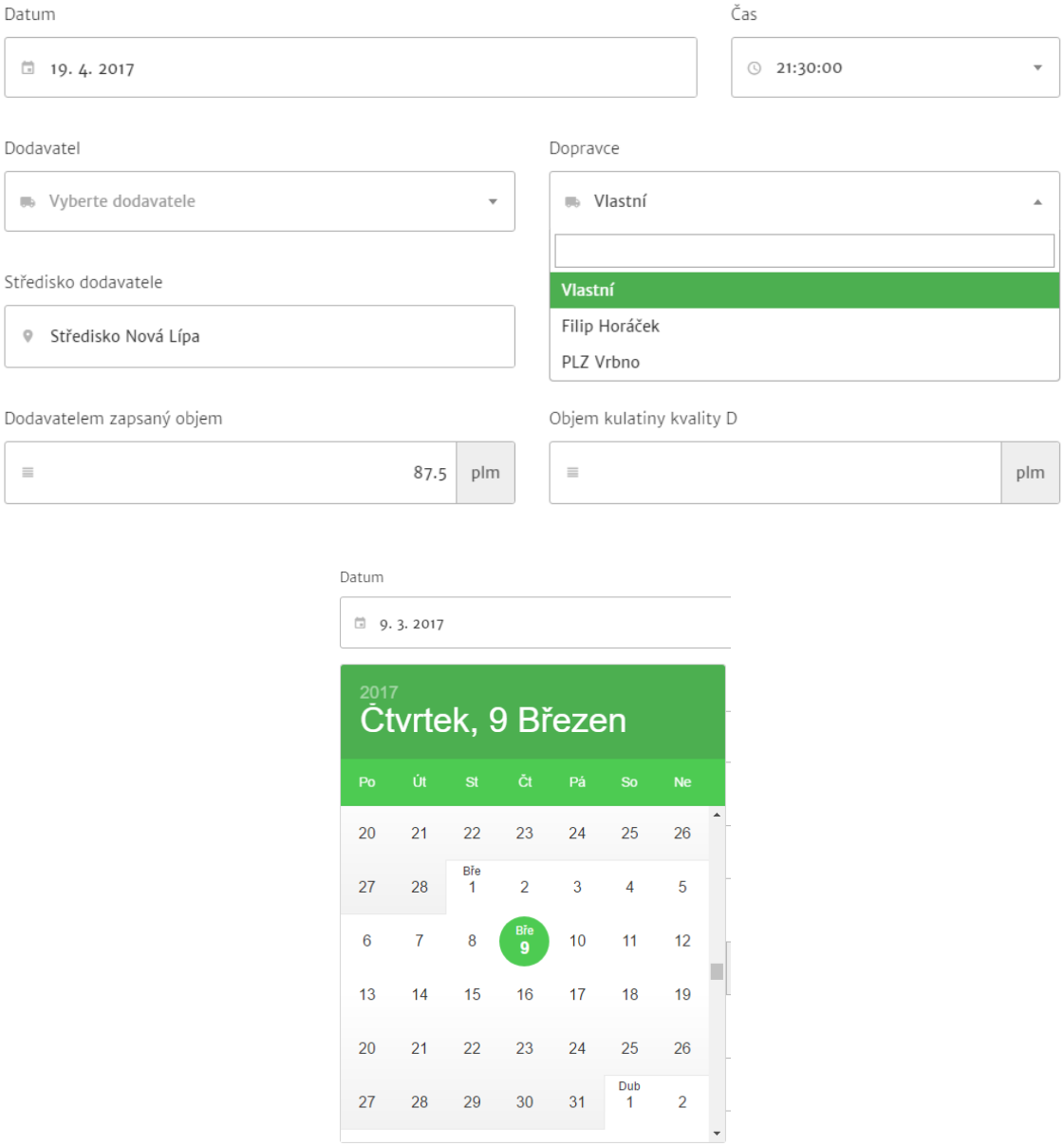
Jak již bylo zmíněno, napříč systémem jsou využity jednoduché tvary, v podstatě všechny grafické prvky jsou tvaru obdélníku se zaoblenými rohy. Mezi nejzákladnější grafický prvek aplikace patří takzvaný Box. Jedná se o kontejner na další prvky uživatelského rozhraní, jež logicky odděluje skupiny prvků, jež patří do stejné jednotky. Příklad Boxu je vidět na následující obrázku, jež zobrazuje výřez obrazovky s formulářem pro změnu uživatelova hesla:



Obrázek 11 - Prvek Box
(Zdroj: Vlastní zpracování)

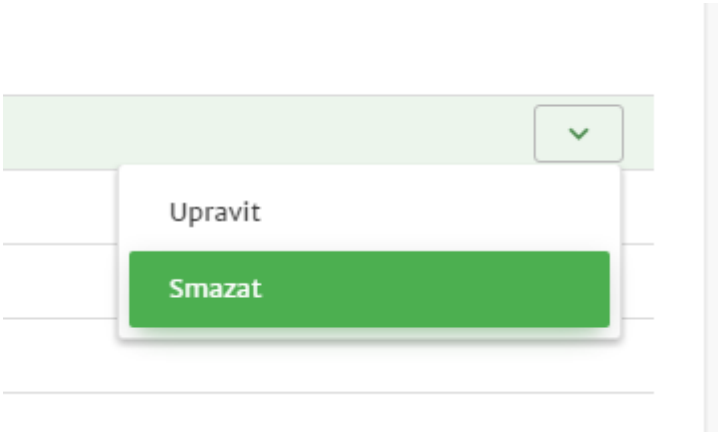
Box obsahuje hlavičku podbarvenou primární barvou a titulek, jež je vždy napsán kapitálkami. Volitelně může hlavička zobrazovat ikonu a případně také podtitulek. Pozadí boxu je bílé a celý box vrhá jemný stín. Rohy boxu i hlavičky jsou zaobleny s rádiusem 3 pixely.

Dalším grafickým prvkem je formulářové pole. Systém obsahuje dva základní prvky: textové pole a roletkový výběr. Tyto prvky jsou dále rozšířeny o varianty pole pro zadání numerické hodnoty a pole pro výběr datumu. Všechny varianty formulářových prvků plus komponenta výběru datumu jsou zobrazeny na následujících obrázcích.



Obrázek 12 - Formulářové prvky
(Zdroj: Vlastní zpracování)

Dalším prvkem používaným napříč systémem je prvek rozbalovacího (kontextového) menu. Toto menu lze zobrazit na libovolném místě aplikace. Při zobrazení překryje ostatní prvky a umožní uživateli zvolit akci. Kliknutím kamkoliv mimo oblast menu je kontextové menu skryto. Typické použití je pro nabídku editace nebo mazání položek.

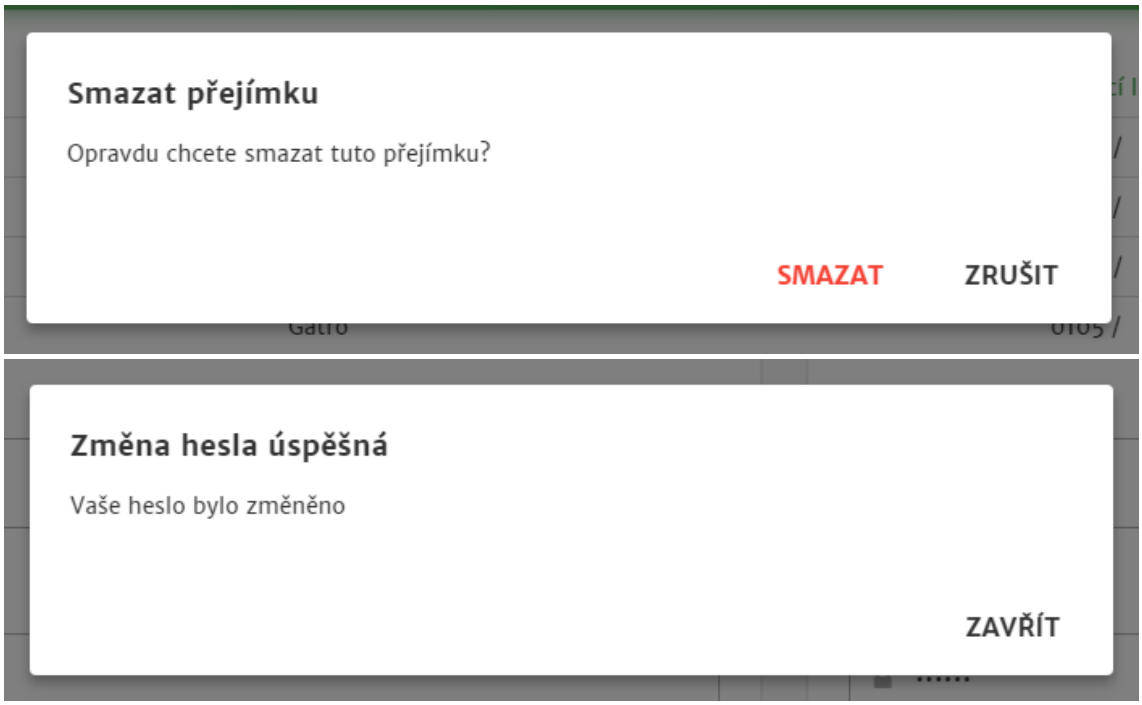


Obrázek 13 – Prvek kontextové menu
(Zdroj: Vlastní zpracování)

Následujícím grafickým prvkem rozšířeným skrze celou aplikaci je prvek tlačítka. Tlačítka slouží k vyvolání určitých akcí, typicky například uložení formuláře či přechod na obrazovku pro vytvoření nového dokumentu. Tlačítko přebírá designový styl tlačítek Material Designu, je tedy obdélníkového tvaru, má 3 pixely zaoblené rohy a text je vyveden v kapitálkách. Barva tlačítka může být buď primární zelená (typicky pro potvrzovací akci) nebo základní šedá (typicky pro odvolání akce).

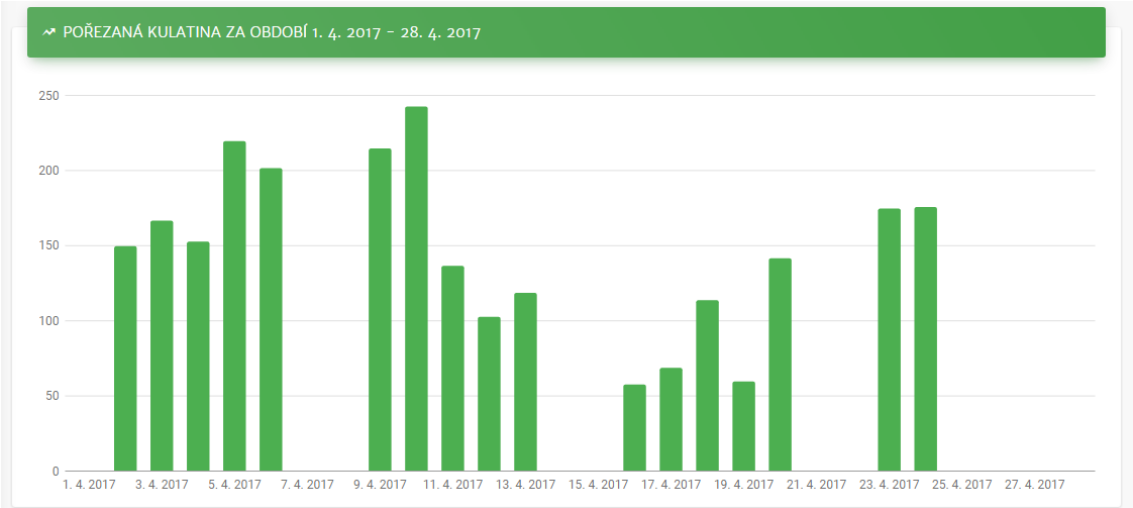
Obrázek 14 - Prvek tlačítko
(Zdroj: Vlastní zpracování)

Posledním grafickým prvkem používaným napříč systémem je dialogové okno. V systému existují dva typy dialogových oken: upozornění a potvrzení akce. Upozornění pouze zobrazí text a nabízí jediné tlačítko, jímž je dialogové okno skryto. Potvrzení naopak dává uživateli vybrat ze dvou nebo více možností. Dialogové okno opět následuje designový styl Material Deisgnu.



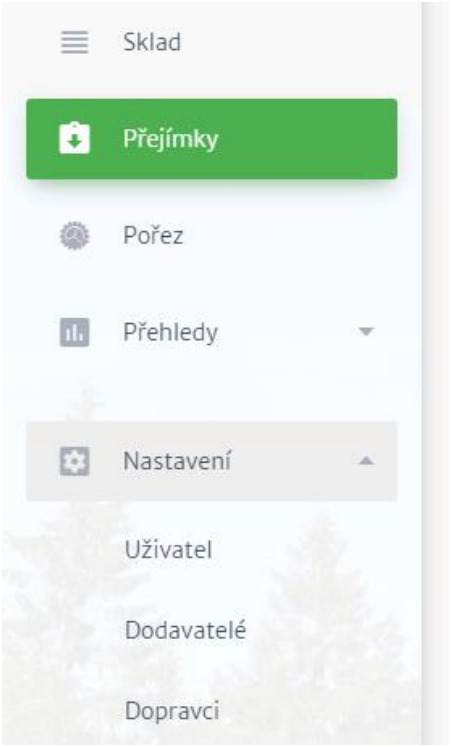
Obrázek 15 - Prvek dialogové okno
(Zdroj: Vlastní zpracování)

Nakonec zbývá doplnit prvek grafického zobrazení datového souboru, tedy graf, jež je využit v modulu Přehledy. V aplikaci jsou využity tři druhy grafů – sloupcový, spojnicový a koláčový. Grafy obsahují stejné barvy jako zbytek aplikace a také implementují Material Design.



Obrázek 16 - Prvek graf
(Zdroj: Vlastní zpracování)

Pro úplnost je třeba zmínit ještě postranní panel na levé straně obrazovky, jež slouží k navigaci mezi jednotlivými moduly a obrazovkami aplikace. Momentálně aktivní modul je v panelu podbarven primární barvou. Panel může obsahovat také sbalovací položky, jež lze skrýt či odkrýt. To je využito u položek Nastavení a Přehledy.



Obrázek 17 - Prvek menu
(Zdroj: Vlastní zpracování)

4.4.3 Funkce aplikace

Poslední podkapitola týkající se klientské aplikace se zabývá funkčností aplikace. Budou zde popsány funkce jednotlivých modulů a kde to bude vhodné, bude popis doplněn vývojovými diagramy. Modul *přejímky* bude navíc popsán detailněji pomocí dalších diagramů. Funkcionalita aplikace bude popsána podle jednotlivých modulů, jež byly definovány v kapitole 4.4.1.

Modul sklad

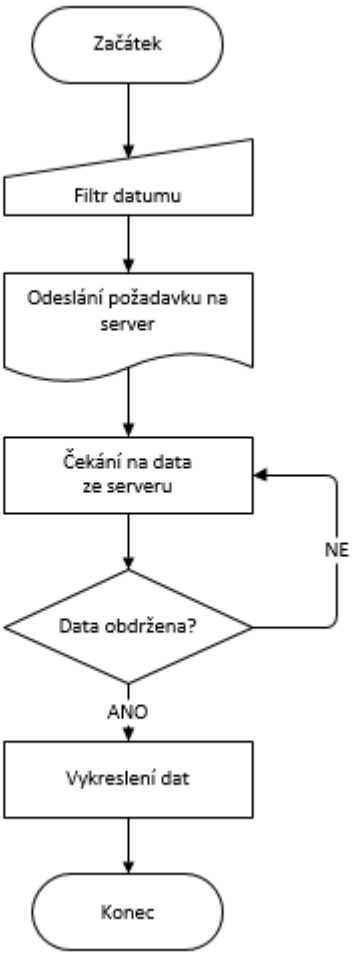
Modul sklad slouží ke sběru dat o přejímkách a pořezech a podávání informací o současné skladové situaci, případně situaci za zvolený rozsah dní. Uživatelská interakce s tím to modulem tedy spočívá pouze ve čtení dat a organizaci filtrování dat. Data obdržená ze serveru jsou již hotová, pro zobrazení připravená data. Samotné agregační operace jsou prováděny na serveru na úrovni databáze pro dosažení co nejlepší výkonnosti a pro odlehčení práce klientské aplikace. Klientská aplikace pouze obdržená data rozdělí do skupin a vykreslí do tabulek. Modul skladu tedy vykonává tyto interakce se severem:

- Načtení seznamu celkových stavů skladu

  `GET /warehouse  parametry {date_from, date_to}`
- Načtení skladových pohybů dle skupin

  `GET /warehouse/grups  parametry {date_from, date_to}`

Vývojový diagram popisující tento proces načtení skladových zásob z pohledu klientské aplikace je na následujícím obrázku.



Obrázek 18 - Vývojový diagram: Zobrazení skladu
(Zdroj: Vlastní zpracování)

Modul přejímky

Modul přejímek vykonává dvě hlavní funkce: získání a zobrazení seznamu přejímek podle zvoleného filtru a tvorba/editace/mazání existujících přejímek. Získání seznamu přejímek je totožným procesem jako získání informací o skladových úrovních,

jež byl popsán v předchozí části. Znovu tedy tento proces detailněji popisován nebude. Obdobným způsobem probíhá také mazání existující přejímky, což je akce vyvolaná uživatelem. Navíc je však uživatel před odesláním požadavku na smazání přejímky vyzván k potvrzení akce pomocí dialogového okna, jež mu dává možnost odvolat potenciálně nechtěnou akci. Zásadním procesem tohoto modelu je přidání nové přejímky (případně editace, jež probíhá obdobně). Uživatel zde nejprve zadává údaje k přejímce, přičemž se v reálném čase přepočítávají celkové stavy přijaté kulatiny, jež jsou vizualizovány v tabulce podle kategorií délky a čepového průměru. Vývojový diagram procesu přidávání nové přejímky je vizualizován na následující straně. Modul přejímek vykonává následující interakce se serverem:

- Načtení seznamu přejímek

  `GET /handlings  parametry {date_from, date_to, supplier}`

  V případě, že doposud nebyl načten seznam dodavatelů, načte také tento

  `GET /suppliers`
- Pokud při otevření obrazovky nové přejímky nebo editace existující doposud nebyly načteny seznamy dodavatelů a dopravců, načte je

  `GET /suppliers`

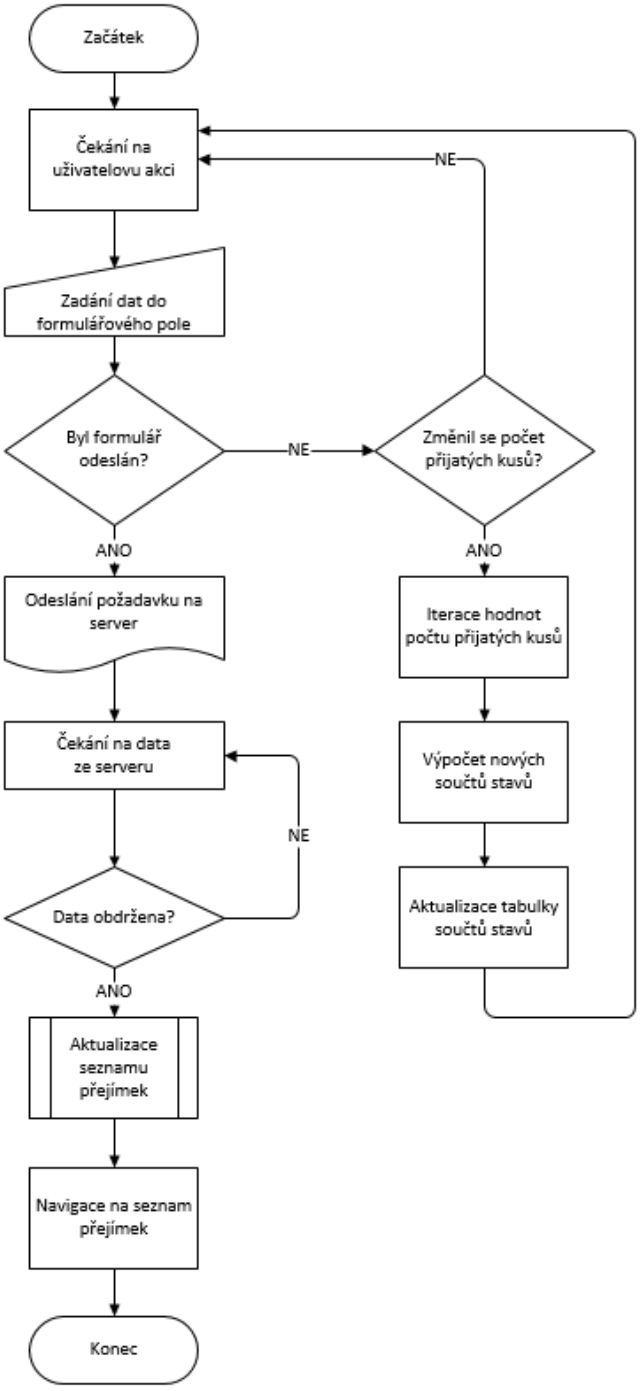
  `GET /haulers`
- Odeslání formuláře nové přejímky

  `POST /handlings  data o přejímce v těle požadavku`
- Načtení existující přejímky k editaci

  `GET /handlings/:id  parametr {id}`
- Uložení editace existující přejímky

  `PUT /handlings/:id  parametr {id}, data o přejímce v těle požadavku`
- Smazání existující přejímky

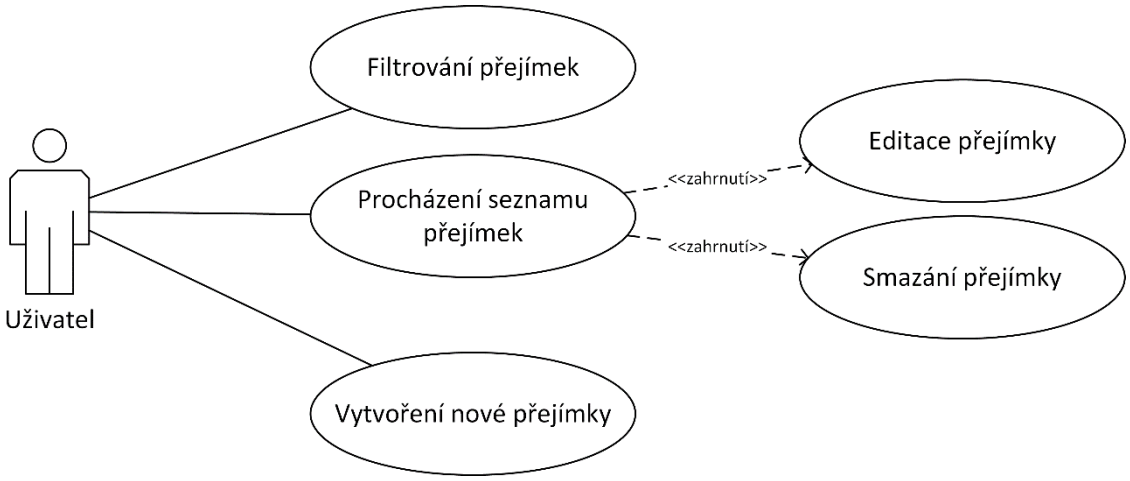
  `DELETE /handlings/:id  parametr {id}`



Obrázek 19 - Vývojový diagram: nová přejímka
(Zdroj: Vlastní zpracování)

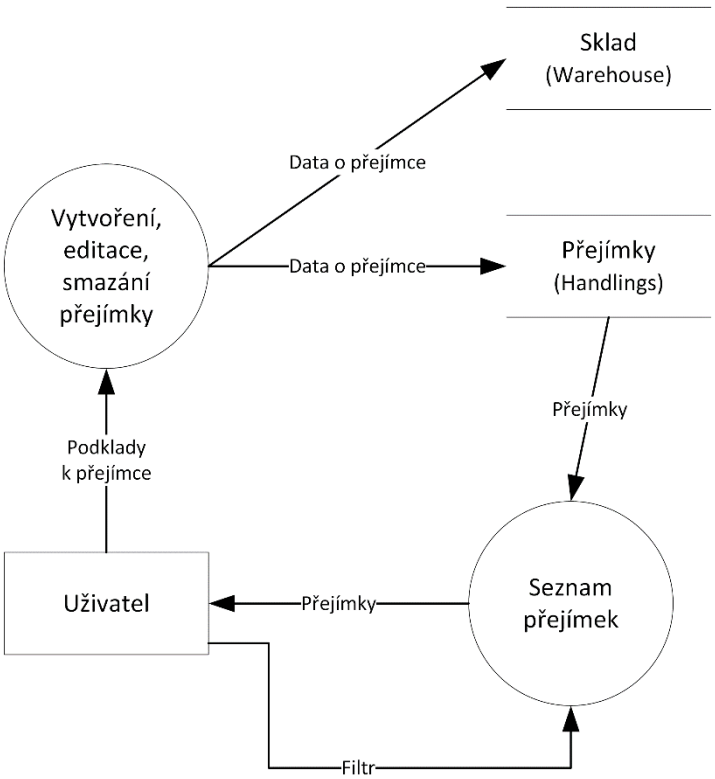
Editace přejímky probíhá stejným způsobem. Rozdíl oproti přidávání nových přejímek spočívá v tom, že u již vytvořených přejímek nelze změnit datum uskutečnění kvůli návaznosti na záznam ve skladu, jež je také časově definovaný.

Z pohledu uživatele lze modul přejímek charakterizovat pomocí následujícího diagramu případů užití.



Obrázek 20 - Diagram případů užití modulu přejímky
(Zdroj: Vlastní zpracování)

Posledním diagramem, jež popisuje modul přejímek, je diagram toku dat. Tento diagram zobrazuje, jak v rámci modulu tečou data mezi uživatelem a systémem.



Obrázek 21 - Diagram toku dat v modulu přejímky
(Zdroj: Vlastní zpracování)

Modul pořez

Stejně jako modul přejímek má i modul pořez dvě hlavní funkce, a to získání a zobrazení seznamu pořezů a tvorba/editace/mazání existujících pořezů. Výpis pořezů se oproti výpis přejímek liší zobrazovanými údaji a také možnostmi filtrování, kdy pořezy lze filtrovat podle datumu pořezu. Vytvoření nového pořezu opět blízce souvisí s procesem přidání přejímky. Liší se hlavně základními údaji, kde pořez definují zejména časové charakteristiky dané směny (celkový čas směny, délka prostojů kvůli případným poruchám, čas stráveny výměnou nástrojů či změnou nastavení výrobních strojů, čas strávený čištěním strojů a ostatní prostoje) a také reálně vyprodukované objemy dřevní hmoty dle různých kategorií jakosti. Zadáním těchto údajů společně s údaji o pořezaných kusech kulatiny je možné vypočítat jak celkové stavy pořezané kulatiny, tak zejména zásadní ukazatel pořezové agendy, a to je plnění normy pořezu. Různé skupiny čepových průměrů (14–17 cm, 18–20 cm atd.) mají definované různé normy pořezaného objemu kulatiny za časový úsek. Norma je dána počtem minut, kolik je potřeba na pořezání jednoho plnometru kulatiny. Celkové stavy a norma plnění pořezu se počítají v reálném čase. Vzorec pro výpočet plnění normy v procentech pro jednu skupinu vypadá následovně:

$$Plnění\ normy\ [\%] = \frac{pořezaný\ objem * norma\ pro\ danou\ skupinu}{čistý\ čas\ řezání} * 100$$

Celková norma vznikne součtem plnění norem za jednotlivé skupiny. Jako příklad bude uveden výpočet normy pro skupinu čepových průměrů 14–17 centimetrů.

$$Norma_{14-17} = 11{,}2\ \text{min / plm}$$
$$Pořezaný\ objem_{14-17} = 38{,}5$$
$$Čistý\ čas\ řezání = 480\ \text{min} = 8\ \text{hod}$$
$$Plnění\ normy_{14-17} = \frac{38{,}5 * 11{,}2}{480} * 100 = 89{,}83\ \%$$

Tento ukazatel je důležitý pro ředitele výroby kvůli sledování výkonnosti pracovníků, ale také posouzení vhodnosti řezaného sortimentu v daný den a v daných podmínkách.

Modul pořez spolupracuje se serverem v rámci následujících komunikačních operací:

- Načtení seznamu pořezů

  ```
  GET /sawings            parametry {date_from, date_to}
  ```

- Odeslání formuláře nového pořezu

  ```
  POST /sawings           data o pořezu v těle požadavku
  ```

- Načtení existujícího pořezu k editaci

  ```
  GET /sawings/:id        parametr {id}
  ```

- Uložení editace existující přejímky

  ```
  PUT /sawings/:id        parametr {id}; data o pořezu v těle
  požadavku
  ```

- Smazání existující přejímky

  ```
  DELETE /sawings/:id     parametr {id}
  ```

Modul přehled

Modul přehled slouží uživatelům aplikace k snadnému zobrazení dat za určité časové období pomocí přehledných grafů. Interakce s uživateli zde tedy, stejně jako v případě modulu skladu, probíhá pouze na úrovni manipulace s filtry časového rozsahu zobrazených dat. Zajímavější procesy probíhají ve fázi přípravy dat pro zobrazení grafů, a to jak na straně serveru, tak na straně klienta. Server provádí při získávání dat pro přehledy různě složité agregační operace nad kolekcemi skladu, manipulace a pořezu, a vrací klientské aplikaci již sečtená data. Klientská aplikace pak data musí naformátovat do podoby vhodné pro zobrazení grafů: provést součty v případě více údajů v jednom dni, provést vyrovnání dat za dny, kdy nenastala změna a ze serveru tak přišla prázdná odpověď pro tyto dny, nastavit popisky dat a rozdělit data do pole osy X a Y. Modul komunikuje se serverem za účelem načtení přehledových dat následovně:

- Načtení statistik skladu

  ```
  GET /stats/warehouse    parametry {date_from, date_to}
  ```

- Načtení statistik přejímek

  ```
  GET /stats/handling     parametry {date_from, date_to}
  ```

- Načtení statistik pořezů

  ```
  GET /stats/sawing       parametry {date_from, date_to}
  ```

Obrázek 22 - Vývojový diagram: zobrazení přehledu
(Zdroj: vlastní zpracování)

Modul nastavení

Modul nastavení slouží ke správě uživatelových profilových informací, jako je jméno a příjmení nebo e-mailová adresa, ke změně uživatelova hesla a ke správě dodavatelů a dopravců. Dodavatelé a dopravci vstupují do tvorby a editace přejímky, kdy každá přejímka má specifikovaného dodavatele a volitelně také dopravce. Správa uživatelova profilu spočívá v jednoduchém formuláři, v němž uživatel může upravit své údaje. Na stejné obrazovce může uživatel také změnit své heslo, jež používá pro přihlášení do systému. Správa dodavatelů a dopravců probíhá na samostatných obrazovkách. Nejprve uživatel vybere dodavatele/dopravce ze seznamu, případně zvolí

volbu Přidat dopravce/Přidat dodavatele a poté vyplní formulář s údaji o dodavateli/dopravci. Z obrazovky přehledu lze dodavatele/dopravce také smazat. Proces přidávání či editace nastavení případně dodavatelů nebo dopravců je tak jednoduchý a přímočarý, že jej není třeba popisovat vývojovým diagramem. Za účelem vykonání popsaných funkcí komunikuje aplikace se serverem následovně:

- Načtení dat o přihlášeném uživateli

  ```
  GET /user
  ```

- Načtení nastavení aplikace

  ```
  GET /settings
  ```

- Načtení seznamu dodavatelů a dopravců

  ```
  GET /suppliers
  GET /haulers
  ```

- Přidání nového dodavatele nebo dopravce

  ```
  POST /suppliers          data o dodavateli v těle požadavku
  POST /haulers            data o dopravci v těle požadavku
  ```

- Načtení existujícího dodavatele nebo dopravce k editaci

  ```
  GET /suppliers/:id       parametr {id}
  GET /haulers/:id         parametr {id}
  ```

- Uložení editace existujícího dodavatele nebo dopravce

  ```
  PUT /suppliers/:id       parametr {id}, data v těle
  PUT /haulers/:id         parametr {id}, data v těle
  ```

- Smazání existujícího dodavatele nebo dopravce

  ```
  DELETE /suppliers/:id    parametr {id}
  DELETE /haulers/:id      parametr {id}
  ```

- Uložení změny uživatelských údajů

  ```
  PUT /users/:id           parametr {id}, data v těle
  ```

- Uložení změny uživatelova hesla

  ```
  PUT /users/:id/password  parametr {id}, data v těle
  ```

4.5 Implementace systému

Jelikož vyvíjený systém je navržen jako náhrada za stávající systém evidence kulatiny ve společnosti, je nutné zvolit vhodnou implementační strategii pro zajištění co nejhladšího procesu implementace. Implementace systému proto bude rozdělena do několika na sebe navazujících fází.

4.5.1 Testování systému

Před zahájením samotného procesu implementace nového systému do provozního prostředí firmy je bezpodmínečně nutné provést důkladné testování celého systému. Samotné testování se skládá z několika oblastí – testování korektní funkčnosti serveru, testování korektní funkčnosti klienta a testování použitelnosti uživatelského rozhraní.

Testování serveru a klientské aplikace

Testování obou částí systému bude probíhat současně. V první fázi budou připraveny testovací scénáře, podle kterých se bude postupovat a zaznamenávat výsledky testů. Testovací scénáře mohou zahrnovat oblasti a podscénáře jako například:

- Vytvoření nové přejímky
    - Se současným datumem
    - S datumem v minulosti
    - Nevyplnění povinných polí
    - Zadání nesmyslných hodnot (text do číselných polí atd.)
- Zobrazení přehledů
    - Přehled za zvolené časové období plus kontrola ekvivalence dat v přehledu a v dané oblasti (skladu, přejímkách, pořezech)
    - Zvolení nesmyslných filtrů (datum od větší než datum do)

Výstupy z těchto testu je třeba pečlivě projít a případné nedostatky opravit. Následně je nutné scénáře, jež neproběhly podle očekávaných výsledků zopakovat.

Testování s reálnými daty

Dalším krokem je testování systému způsobem, jež by se co nejvíce přiblížil každodennímu využívání ve firmě. To znamená obstarání datového souboru reálných dat z firmy a simulace každodenního používání. V případě vyvíjeného systému budou využita data o přijatém a pořezaném zboží za jeden měsíc. Tato data budou vložena do

systému a proběhne kontrola, zda uložené a vypočítané stavy odpovídají hodnotám daného měsíce ve starém systému. Díky tomuto testování lze odhalit na první pohled neviditelné chyby, jež se při práci s náhodnými testovacími daty nemusí projevit.

Testování použitelnosti uživatelského rozhraní

V tomto kroku testování již budou do procesu testování zapojeni i externí osoby. Představitelé uživatelů systému ve firmě budou pozváni k seznámení se se systémem a k vyzkoušení si vlastní práce s (téměř) finálním produktem. Tento krok je důležitý, jelikož navzdory spolupráci s představiteli uživatelů z řad firmy v průběhu celého vývoje systému byly některé, zejména vizuální, prvky vyvíjeny samostatně a bez možnosti okamžité zpětné vazby od představitelů firmy. Bude tedy nutné, aby si uživatelé systém osahali, takzvaně „proklikali“, a podali zpětnou vazbu. Na základě zpětné vazby bude v případě nutnosti přistoupeno k drobným modifikacím uživatelského rozhraní systému tak, aby byli budoucí uživatelé maximálně spokojeni s pohodlností práce.

4.5.2 Školení uživatelů

Dalším nesmírně důležitým krokem je školení uživatelů v používání systému. Před nasazením systému do firemního prostředí je nutné, aby byli všichni jeho uživatelé detailně seznámeni s veškerou funkcionalitou systému a procesy, jež systém pokrývá. Jelikož firma S T A R P s.r.o. je malá firma a uživatelská základna systému čítá pouze jednotky uživatelů, bude celý proces školení značně zjednodušený díky možnosti osobního přístupu ke každému uživateli.

4.5.3 Nasazení ve firemním prostředí

Poslední fází implementace je samotné nasazení systému ve firemním prostředí. Jelikož systém slouží jako náhrada stávajícího systému, jež nyní obsahuje pro firmu důležitá data, je nutné dbát velký důraz na zodpovědné dodržení všech následujících procesů.

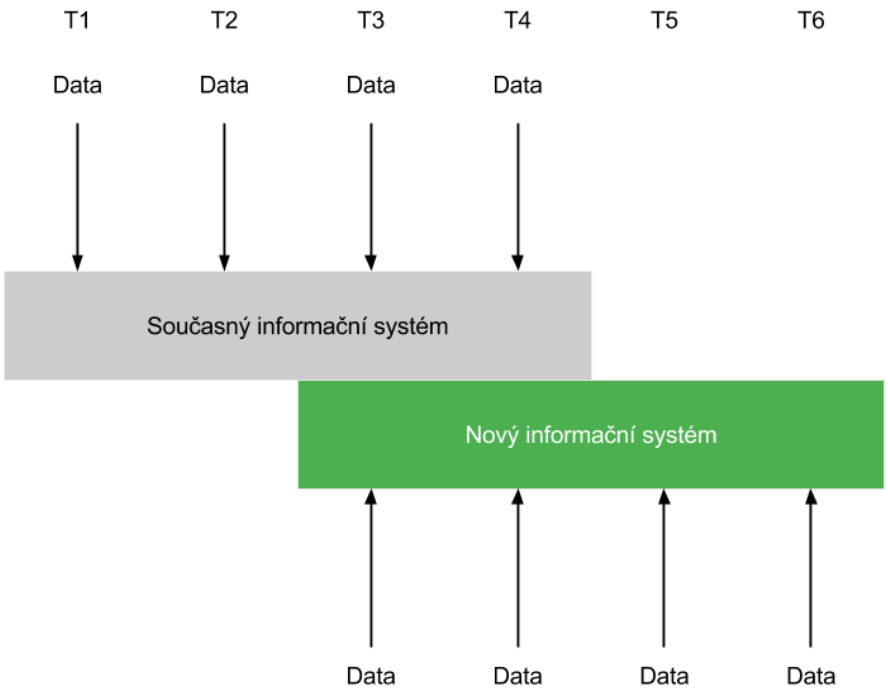
Migrace dat

Prvním krokem je migrace stávajících dat. Současný systém evidence skladu eviduje vždy pouze data za aktuální měsíc plus celkové stavy skladu za minulé období. To do určité míry zjednodušuje proces migrace dat do nového systému, jelikož bude nutné přenést data pouze za posledních maximálně 31 dní plus celkový stav z minulého období.

Zároveň však vznikne v sytému nárazová odchylka od denních průměrů, jež bude zejména v modulu přehledu způsobovat z počátku nepřesnosti (které však lze odstranit vhodným zvolením filtrů rozsahu datumu). Migrace tedy proběhne formou „inicializační přejímky“, jež zavede do skladu současné stavy skladu v jednotlivých kategoriích za minulé období. Následně proběhne migrace dat (přijatého a pořezaného zboží) za každý den minulého měsíce – tato data již odpovídají reálné situaci, včetně rozdělení přijatých a pořezaných kusů přesně podle odpovídajících kategorií. Po kontrole a porovnání dat se současným systémem bude nový systém připraven na nasazení do ostrého provozu.

Nasazení do ostrého provozu

Ačkoliv v této fázi byl systém již několikrát testován, bude přistoupeno k opatrnější strategii nasazení do provozu – konkrétně bude zvolena souběžná strategie. Po dobu zvoleného chráněného období budou oba systémy používány současně – tedy data budou vkládána jak do starého, tak i do nového informačního systému. Ačkoliv to po dobu zvoleného období (v tomto případě bylo zvoleno období 14 dní) sníží produktivitu práce kvůli nutnosti zadávat stejná data na dvě různá místa, umožní to okamžitou kontrolu chování nového sytému a případně odhalení nesprávného chování. Průběh nasazení do provozu je vizualizován na následujícím obrázku.



Obrázek 23 - Implementační strategie
(Zdroj: Vlastní zpracování)

4.6 Ekonomické zhodnocení

Tato kapitola zhodnotí celý projekt výměny stávajícího informačního systému za nový systém pro evidenci kulatiny z hlediska nákladů a přínosů. Ekonomické hodnocení je nedílnou součástí každého projektu, jelikož pro firmu nemá smysl pořizovat něco, co svými náklady převýší možné výnosy.

4.6.1 Časová náročnost projektu

Prvním údajem potřebným pro stanovení nákladů a přínosů projektu výměny informačního systému je časová náročnost projektu. Z údajů uvedených v této kapitole následně bude vycházet kvantifikace nákladů vynaložených na realizaci projektu. Celý projekt lze rozdělit do několika oddělených a na sebe navazujících úseků.

- Návrh architektury, datového a funkčního modelu
    - Návrh architektury a datového modelu 4 hodiny
    - Návrh funkčního modelu 4 hodiny
- Vývoj serverové části aplikace
    - Návrh REST architektury 3 hodiny
    - Programování funkční části 17 hodin
    - Implementace OAuth 2.0 protokolu 6 hodin
- Vývoj klientské částí
    - Návrh grafiky a uživatelského rozhraní 8 hodin
    - Vývoj interaktivního uživatel. rozhraní 5 hodin
    - Vývoj funkcionality aplikace 28 hodin
- Implementace
    - Testování systému 8 hodin
    - Školení uživatelů 4 hodiny
    - Nasazení do provozu 8 hodin
- **Celkem** **95 hodin**

4.6.2 Náklady

Náklady na pořízení systému a náklady plynoucí z jeho provozu jsou velmi důležitou položkou vstupující do rozhodování, zda takový systém pořídit či nikoliv. Existuje mnoho strategií, jak přistoupit k řízení nákladů spojených s pořízením nového informačního systému ve firmě. V případě firmy S T A R P s.r.o. a jejího projektu pořízení nového systému pro evidenci kulatiny bylo rozhodnuto o realizaci prostřednictvím vlastního řešení nového systému, a to zejména z důvodu, že celé programátorské práce obstaral autor této práce bez požadavku peněžní odměny za vykonanou činnost. Přesto je nutné náklady kvantitativně vyjádřit, a proto bude pro výpočet nákladů na práce vývojáře systému využita hodinová mzda, která dle vlastních zkušeností autora odpovídá průměrné hodinové mzdě programátora webových aplikací na volné noze – tedy 300 Kč za hodinu. Dále je potřeba zahrnout náklady, jež vyjadřují čas zaměstnanců firmy, který strávili konzultacemi, školením, testováním a migrací dat. Tato hodnota nákladů byla odhadnuta a poskytnuta firmou. Další položkou nákladů jsou náklady na hardwarové prostředky. V kapitole analýzy současného stavu byla popsána úroveň hardwarového vybavení firmy, jež je pro potřeby vyvinutého systému dostatečná, a proto není třeba pořizovat nové hardwarové prostředky – tato položka je tedy nulová. Poslední nákladovou položkou jsou náklady na provoz serverového prostředí, jež bude realizováno pomocí pronájmu virtuálního privátního serveru (VPS). Celkové vyčíslení nákladů je tedy zobrazeno v následující tabulce.

Tabulka 20 - Celkové náklady na realizaci
(Zdroj: Vlastní zpracování)

Položka	Počet	Cena
Náklady na vývoj a implementaci	95 hodin, 300 Kč/hod	28 500 Kč
Náklady na zaškolení zaměstnanců	10 hodin, 100 Kč/hod	1 000 Kč
Náklady na pořízení hardware	0 ks	0 Kč
Náklady na provoz VPS	1 rok	2500 Kč
Celkem		**32 000 Kč**

4.6.3 Přínosy

V této části budou zhodnoceny přínosy implementace nového informačního systému do firemního provozu. Nejprve bude uveden výstup SWOT analýzy nového

řešení informačního systému, jež ukazuje – zejména v porovnání s výstupem SWOT analýzy současného řešení (kapitola 3.9.2 na straně 49) – výrazné zlepšení situace.

Tabulka 21 - Výstup SWOT analýzy nového systému
(Zdroj: Vlastní zpracování)

Silné stránky	Slabé stránky
• Řešení přizpůsobené požadavkům firmy • Zajištění uchování historie a integrity dat • Statistická a analytická část přehledů • Správa uživatelů systému a přístupu • Přehledné a příjemné uživatelské prostředí • Provoz v cloudu – přístup k systému i mimo kancelář firmy	• Nové řešení, na které uživatelé nejsou zvyklí • Provoz v cloudu – nutnost konektivity k internetu
Příležitosti	**Hrozby**
• Vytvoření offline režimu aplikace s následnou automatickou synchronizací dat • Vytvoření mobilní části systému pro odbourání přepisování dat z papírové podoby	• Ztráta konektivity k internetu a omezení možnosti práce se systémem

V následující části jsou uvedeny další přínosy nového řešení informačního systému. Přínosy leží jak v rovině kvantifikovatelných přínosů (například ušetřený čas), tak v rovině kvalitativní (například větší pohodlnost používání). Následující výčet obsahuje výčet kvantitativně vyjádřených a kvalitativně vyjádřených přínosů, jež byly identifikovány společně s pracovníky firmy a uživateli systému v provozním prostředí firmy.

Kvantitativně vyjádřené přínosy

- **Úspora času při každodenní práci:** díky přehlednému uživatelskému prostředí, snadné orientaci v systému a uživatelsky příjemným ovládacím prvkům systému dojde k odhadovanému ušetření času v řádu desítek minut denně. Ač se to může zdát jako zanedbatelná hodnota, za týden se jedná o hodiny a za měsíc lze hovořit o úspoře v řádu tisíců korun.
- **Úspora času při měsíční uzávěrce:** jelikož nový systém již uchovává data za minulé měsíce bez nutnosti ruční archivace předchozího měsíce a přípravy

sešitů na další měsíc, ušetří ředitel výroby, jež tuto agendu zastával, na konci každého měsíce průměrně 2 hodiny času.

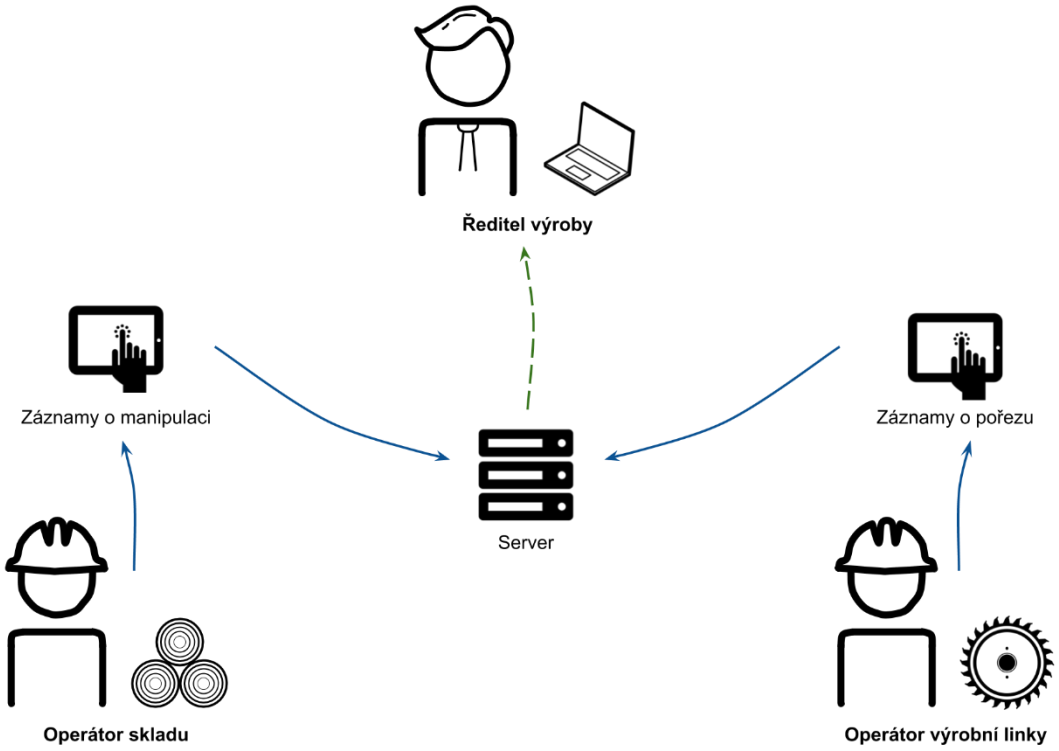
Kvalitativně vyjádřené přínosy

- **Lepší možnost plánování:** díky možnosti zobrazení přehledů a údajů za minulé období lze lépe plánovat objednávky nového zboží a určovat strukturu sortimentu řezaného zboží.
- **Pohodlnější práce:** jelikož systém byl vyvíjen s ohledem na co nejlepší uživatelskou zkušenost a pohodlnost použití, očekává se, že uživatelé budou s používáním systému spokojenější, než jsou se stávajícím systémem.
- **Vyšší bezpečnost dat:** protože starý systém byl postaven na sešitech MS Excel, data nebyla pořádně zálohována a integrita dat závisela na správném opsání údajů z minulého časového období, byla i celková bezpečnost dat ohrožena. Nový systém přinese značné zvýšení bezpečnosti dat a automatické zajištění zabezpečení integrity dat.

4.7 Vize do budoucna

Celý informační systém byl vytvářen s ohledem na univerzálnost a možnost budoucího rozšíření funkcionality či možností využití. Vize do budoucna spočívá ve vývoji systému v duchu co nejvyšší automatizace a s tím spojeným úbytkem redundance práce, jež se nyní v procesech firmy vyskytuje (viz kapitola 3.9.1, strana 47). Zároveň bude v blízké budoucnosti nutné přidat do systému offline funkcionalitu, jež zajistí možnost zaznamenávat data do systému i při ztrátě konektivity k internetu, a samozřejmě následnou automatickou synchronizaci dat po opětovném připojení. Výraznější změnou, jež přímo ovlivní procesy ve firmě, však bude plán delšího časového horizontu, jež počítá s vyvinutím mobilní klientské aplikace, jež bude komunikovat prostřednictvím stejného REST rozhraní se serverem. Tato mobilní aplikace bude sloužit pracovníkům výroby, kteří provádějí manipulaci a evidenci přijatého dřeva a evidenci pořezaného materiálu. Namísto současného modelu, kdy tito pracovníci zmíněné údaje zapisují do papírového sešitu a ředitel výroby následně přepisuje tato data do informačního systému, tak pracovníci budou data zapisovat přímo do systému pomocí mobilní aplikace. Ředitel výroby pak bude pouze kontrolovat zapsaná data. Tímto způsobem ušetří ředitel výroby až hodinu času denně, kterou by jinak strávil přepisováním údajů ze sešitů do

informačního systému. Následující obrázek zobrazuje situaci, kdy je systém ve stavu popsaném výše. Ředitel výroby zde již zastává pouze kontrolní funkci, veškeré záznamy vkládají do systému samotní pracovníci výroby.



Obrázek 24 - Používání systému po zapojení mobilní aplikace
(Zdroj: Vlastní zpracování)

Závěr

Cílem této práce bylo navrhnout a vyvinout systém pro evidenci kulatiny pro firmu S T A R P s.r.o. Tento systém byl navržen a vyvíjen na základě provedených analýz, z jejichž závěrů a z podkladů firmy vznikly požadavky na realizaci nového řešení informačního systému. První část práce se zabývala popsáním teoretických východisek, jež čtenáře seznámily s pojmy využívanými v této práci a s technologiemi, jež byly pro tvorbu využity. Nejprve byl popsán programovací jazyk, jež byl k tvorbě nového systému využit a z něhož vychází všechny další technologie. Následoval popis architektury klient-server, na který navazoval popis technologií využitých pro tvorbu serverové a klientské části systému. Následovala kapitola představení společnosti, pro kterou je systém vyvíjen, představení klíčových procesů s důrazem na procesy, jež má nový systém pokrývat. Následovaly analýzy současného řešení informačního systému, jejichž výstupem bylo rozhodnutí, že nový informační systém je potřebný a daly východisko k vlastní realizaci nového řešení informačního systému. Poslední částí kapitoly analýzy současného stavu bylo představení požadavků firmy na nový systém. Poslední a nejdůležitější kapitolou této práce, jež vychází z předchozích dvou, byla kapitola vlastního řešení realizace stanovených cílů. Kapitola začala dekompozicí požadavků firmy, z níž vzešel prvotní návrh struktury aplikace. Z tohoto návrhu dále vycházely všechny ostatní návrhy, od celkové architektury systému přes datové a funkční modelování serverové a klientské aplikace až po představení návrhu uživatelského rozhraní klientské aplikace. Nakonec této práce bylo provedeno ekonomické zhodnocení celého projektu výměny stávajícího řešení informačního systému za nové, kde byly vyčísleny náklady na realizaci s uvážením časového rámce projektu a představeny přínosy nového systému pro firmu. Pro autora práce pak přínosy její realizace tkví zejména v osvojení si znalostí v oblasti pokročilých technologií vývoje moderních webových aplikací. Tyto znalosti jsou v dnešní době velmi žádané na trhu práce.

Seznam použitých zdrojů

BABEL. *Babel* [online]. c2015-2017 [cit. 2017-03-28]. Dostupné z: http://babeljs.io/

BASL, Josef a Roman BLAŽÍČEK. *Podnikové informační systémy. Podnik v informační společnosti.* 1. vyd. Praha: Grada, 2008. 283 s. ISBN 978-80-247-2279-5.

CHODOROW, Kristina a Michael DIROLF. *MongoDB: the definitive guide*. Beijing: O'Reilly, 2010. ISBN 14-493-8156-1.

FACEBOOK INC. *Components and Props – React* [online] c2017 [cit. 2017-03-25]. Dostupné z: https://facebook.github.io/react/

FINK, Gil a Ido FLATOW. *Pro single page application development: using Backbone.js and ASP.NET*. Apress, 2014. ISBN 978-1-4302-6674-7.

GACKENHEIMER, Cory. *Introduction to React*. Apress, 2015. ISBN 978-148-4212-462.

IHRIG, Colin. *Pro node.js for developers*. Apress, 2013. ISBN 978-1-4302-5861-2.

KOCH, Miloš. Zefis – HOS8 – posouzení vyváženosti informačního systému. *Zefis.cz* [online]. c2014 [cit. 2017-01-16]. Dostupné z: *http://www.zefis.cz/*

MOLNÁR, Zdeněk. *Automatizované informační systémy.* Praha: Strojní fakulta ČVUT, 2000. 126 s. ISBN 80-01-02269-2.

MOLNÁR, Zdeněk. *Efektivnost informačních systémů*. Praha: Grada Publishing, 2000. 142 s. ISBN 80-7169-410-X.

PAUTASSO, Cesare, Erik WILDE a Rosa ALARCÓN. *REST: advanced research topics and practical applications*. New York: Springer-Verlag, 2014. ISBN 978-146-1492-986.

ŘEPA, Václav. Analýza a návrh informačních systémů. Praha: Ekopress, 1999. 403 s. ISBN 80-86119-13-0.

SATERNOS, Casimir, Simon ST. LAURENT a Allyson MACDONALD. *Client-server web apps with JavaScript and Java*. O'Reilly Media, 2014. ISBN 978-1-4493-6933-0.

OLORUNTOBA, Samuel. *Getting Started with Webpack: Module Bundling Magic*. In: Scotch.io [online]. 2016 [cit. 2017-03-03]. Dostupné z: https://scotch.io/tutorials/getting-started-with-webpack-module-bundling-magic

SIRIWARDENA, Prabath. *Advanced api security: securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE*. Apress, 2014. ISBN 978-1-4302-6817-8.

SODOMKA, Petr a Hana KLČOVÁ. Informační systémy v podnikové praxi. 2. aktualiz. a rozš. vyd. Brno: Computer Press, 2010. 501 s. ISBN 978-80-251-2878-7.

SOSINSKY, Barrie. *Networking bible.* New Jersey: Wiley, 2013. ISBN 978-047-0543-429.

TECHOPEDIA INC. *What is Javascript (JS)?* [online]. c2017 [cit. 2017-02-27]. Dostupné z: www.techopedia.com

W3TECHS. *Apache vs. Node.js usage statistics*. [online]. c2009-2017 [cit. 2017-03-15] Dostupné z: www.w3techs.com

WEB EDUCATION COMMUNITY GROUP. *A Short History of JavaScript*. [online]. c2017 [cit. 2017-03-10] Dostupné z: www.w3.org

WEBPACK. *webpack* [online]. c2017 [cit. 2017-03-13] Dostupné z: https://webpack.github.io/

Seznam použitých obrázků

Seznam použitých tabulek